# abc INFORMATIQUE

N° 17

COURS D'INFORMATIQUE PRATIQUE ET FAMILIALE

L'ergonomie informatique

Suivez la piste

Le TO 7 de Thomson

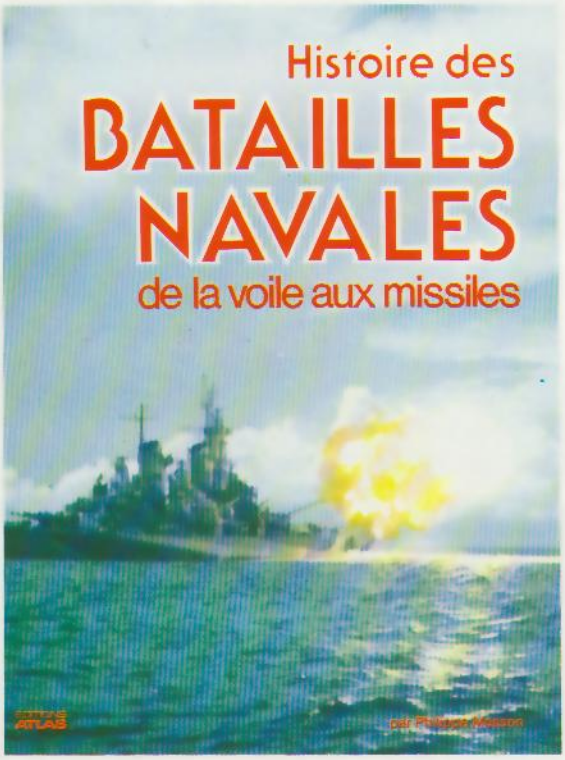
Les expansions du ZX-81

EDITIONS ATLAS

M6062-17-12F 85 FB - 3,80 FS - $ 1.95

EDITIONS ATLAS EDITIONS ATLAS EDITIONS ATLAS EDITIONS ATLAS

Dans toutes les librairies




## Histoire des batailles navales
de la voile aux missiles

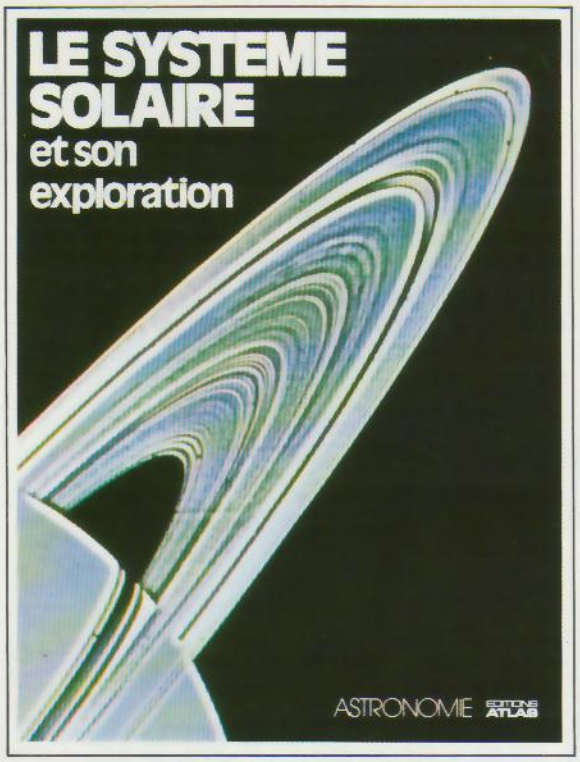
Des duels de l'Antiquité au conflit des Malouines, cet ouvrage retrace vingt-cinq siècles de guerre sur mer.
Ce livre, vivant et précis, illustré de documents anciens et récents sur les navires, les hommes et les opérations qui ont marqué le contrôle de la mer, restitue aux conflits maritimes leur place primordiale dans l'histoire de la guerre.

*Un volume relié, sous jaquette illustrée. 224 pages. 97 photos, 2 dessins et 2 cartes en couleurs. 110 photos et 4 cartes en noir et blanc.*
*Format : 22,5 × 29,5 cm.*

EDITIONS ATLAS EDITIONS ATLAS EDITIONS ATLAS EDITIONS ATLAS

EDITIONS ATLAS EDITIONS ATLAS EDITIONS ATLAS EDITIONS ATLAS

Dans toutes les librairies




## Le système solaire et son exploration

L'aventure de l'observation du ciel commence par l'exploration de notre système solaire. C'est à la découverte des planètes, des satellites, des astéroïdes, des comètes et des météorites qui le composent que vous convie cet ouvrage. Il permet aussi de revivre l'épopée spatiale qui marque la seconde moitié du XXe siècle.
Véritable promenade par les chemins de l'Univers, *Le Système solaire et son exploration* dévoile un monde fascinant.

*Un volume relié.*
*Couverture illustrée. 200 pages.*
*144 photos, 74 dessins et schémas en couleurs.*
*13 photos en noir et blanc.*
*Format : 22 × 28,5 cm.*

EDITIONS ATLAS EDITIONS ATLAS EDITIONS ATLAS EDITIONS ATLAS

aBc INFORMATIQUE
cours d'informatique pratique et familiale
EDITIONS ATLAS

**Édité** par ÉDITIONS ATLAS s.a., tour Maine-Montparnasse, 33, avenue du Maine, 75755 Paris *Cedex 15*. Tél. : 538-52-70.

Belgique : ÉDITIONS ATLEN s.a., Bruxelles.

Canada : ÉDITIONS ATLAS CANADA Ltée, Montréal Nord.

Suisse : FINABUCH s.a., ÉDITIONS TRANSALPINES, Mezzovico.

**Réalisé** par EDENA s.a., 29, boulevard Edgar-Quinet, 75014 Paris. Tél. : 320-15-01.

*Direction éditoriale* : J.-Fr. Gautier. *Service technique et artistique* : F. Givone et J.-Cl. Bernar. *Iconographie* : J. Pierre. *Correction* : B. Noël.

**VENTE**

Les numéros parus peuvent être obtenus chez les marchands de journaux ou, à défaut, chez les éditeurs, au prix en vigueur au moment de la commande. Ils resteront en principe disponibles pendant six mois après la parution du dernier fascicule de la série. (Pour toute commande par lettre, joindre à votre courrier le règlement, majoré de 10 % de frais de port.)

Pour la France, s'adresser à ÉDITIONS ATLAS, tour Maine-Montparnasse, 33, avenue du Maine, 75755 Paris *Cedex 15*. Tél. : 538-52-70.

Pour les autres pays, s'adresser aux éditeurs indiqués ci-dessous.

**SOUSCRIPTION**

Les lecteurs désirant souscrire à l'ensemble de cet ouvrage peuvent s'adresser à :

France : DIFFUSION ATLAS, 3, rue de la Taye, 28110 Lucé. Tél. : (37) 35-40-23.

Belgique : ÉDITIONS ATLEN s.a., 55, avenue Huart-Hamoir, 1030 Bruxelles. Tél. : (02) 242-39-00. Banque Bruxelles-Lambert, compte n° 310-0018465-24 Bruxelles.

Canada : ÉDITIONS ATLAS CANADA Ltée, 11450 boulevard Albert-Hudon, Montréal Nord, H 1G 3J9.

Suisse : FINABUCH s.a., ÉDITIONS TRANSALPINES, zona industriale 6849 Mezzovico-Lugano. Tél. : (091) 95-27-44.

**RELIEZ VOS FASCICULES**

**Des reliures mobiles, permettant de relier 12 fascicules, seront en vente en permanence chez votre marchand de journaux.**

**ATTENTION : ces reliures, présentées sans numérotation, sont valables indifféremment pour tous les volumes de votre collection. Vous les numéroterez vous-même à l'aide du décalque qui est fourni (avec les instructions nécessaires) dans chaque reliure.**

**En vente tous les vendredis. Volume II, n° 17.**

ABC INFORMATIQUE est réalisé avec la collaboration de Trystan Mordrel *(secrétariat de rédaction)*, J.-P. Bourcier *(coordination)*, S.I.-André Larochelle *(traduction)*, Ghislaine Goullier *(fabrication)*, Marie-Claire Jacquet *(iconographie)*, Patrick Boman *(correction)*.
Crédit photographique, couverture : Photo Orbis.

Directeur de la publication : Paul Bernabeu. Imprimé en Italie par I.G.D.A., Officine Grafiche, Novara. Distribution en France : N.M.P.P. Tax. Dépôt légal : avril 1984. 27844. Dépôt légal en Belgique : D/84/2783/27.

© Orbis Publishing Ltd., London.
© Éditions Atlas, Paris, 1984.

A NOS LECTEURS

En achetant chaque semaine votre fascicule chez le même marchand de journaux, vous serez certain d'être immédiatement servi, en nous facilitant la précision de la distribution. Nous vous en remercions d'avance.

Les Éditions Atlas

# Confort au travail

**L'ergonomie est la science qui cherche à améliorer les conditions de travail. En informatique, les travaux se sont concentrés sur le clavier et sur l'écran.**

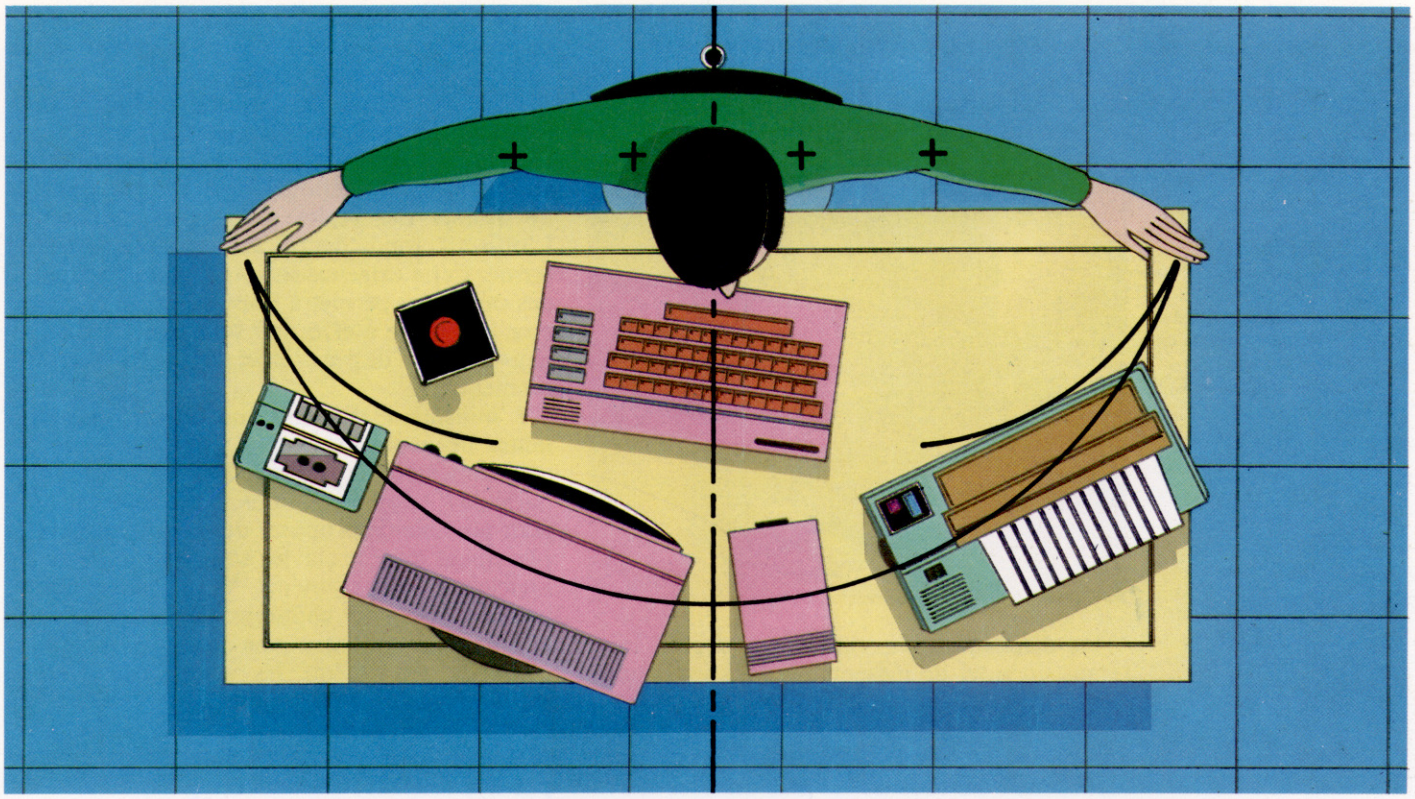
Deux aspects importants ressortent lors de la conception d'un matériel : l'esthétique — la beauté dans la forme —, et l'ergonomie — l'étude des relations entre les travailleurs et leurs outils. Quelle que soit l'efficacité de fonctionnement d'un équipement, il ne sera pas agréable à utiliser s'il est laid. Dans le même esprit, l'environnement du travail doit être dans son ensemble confortable.

Parmi les facteurs dont on tient compte lors de l'achat d'un micro-ordinateur, la qualité de l'ergonomie est de moindre importance que le prix et le rendement de la machine. Mais il est impossible de la négliger. D'abord travaillez-vous sur quelque chose qui ressemble à un poste de travail de bureau, c'est-à-dire sur un plan de travail d'une surface suffisante et à une hauteur adéquate? Ou ne faites-vous simplement que connecter votre ordinateur domestique à votre téléviseur en plaçant la machine sur vos genoux, ou pis encore, posez-vous l'ordinateur sur le sol en face du téléviseur?

La programmation informatique est déjà assez compliquée pour qu'on ne la rende pas encore plus difficile en travaillant dans un environnement totalement inadéquat. Pour créer un poste de travail plus confortable, examinons d'abord ce qui peut être fait au niveau de l'écran. Si vous utilisez un téléviseur domestique, vous ne profitez pas des récents progrès qui aident à réduire ou à éliminer les reflets sur l'écran. On emploie, en effet, des filtres pour réduire les reflets et des affichages au phosphore pour obtenir une meilleure luminosité. Mais vous pouvez vous-même améliorer la qualité de l'image en plaçant un filtre devant l'écran. De simples filtres colorés sont faciles à trouver (un filtre polarisant permet d'éliminer les reflets). Il est ainsi possible d'obtenir un contraste élevé à des niveaux de brillance assez bas et d'éviter de fatiguer les yeux inutilement.

L'intensité de l'éclairage ambiant est aussi importante. Lors d'un travail nocturne, il est préférable d'utiliser une lampe de bureau à faible intensité qui éclaire le clavier et les documents sur lesquels vous travaillez, mais qui laisse l'écran dans une relative obscurité. La distance entre les yeux et l'écran joue également un rôle important (la longueur d'un bras doit les séparer). L'écran inclinable est préférable, afin de pouvoir le régler perpendiculairement par rapport à votre ligne de vue. Cette condition

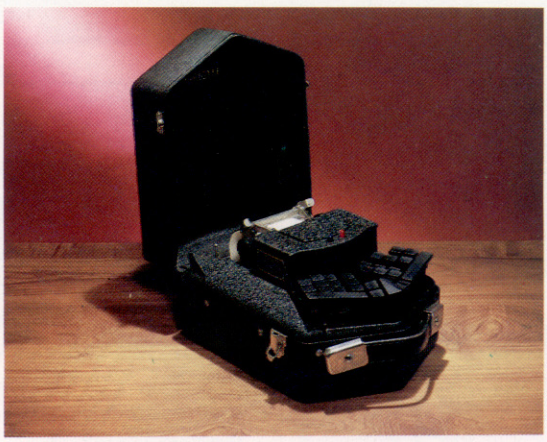
**Langage du corps**
Même si le corps humain varie au niveau des dimensions et des formes, les proportions demeurent assez constantes, comme tout étudiant en dessin peut le constater. L'ergonomie utilise cette constante pour définir des règles générales d'environnement de travail. En informatique domestique, ces règles suggèrent que l'écran soit à une distance égale à la longueur d'un bras (pour minimiser le travail de mise au point de l'œil lorsque celui-ci doit passer fréquemment du document source à l'écran). Ces règles définissent également la position du clavier. (Cl. Roy Ingram.)



peut facilement être obtenue en plaçant un livre ou deux sous l'avant du moniteur. Cependant, il reste possible que votre propre reflet apparaisse sur l'écran.

Après avoir rendu l'écran plus confortable, examinons maintenant la disposition matérielle et les caractéristiques du clavier. Les facteurs les plus importants sont la hauteur des touches au-dessus du plan de travail et l'angle que forment les rangées de touches l'une par rapport à l'autre. L'idéal est que le clavier soit assez bas pour que les poignets et les avant-bras de l'opérateur puissent reposer à plat sur le bureau. Son inclinaison doit être réglable. Malheureusement, peu d'ordinateurs domestiques offrent ces propriétés. La série ZX de Sinclair, l'Oric-1, le Jupiter et le Thomson TO 7 possèdent des claviers assez bas, mais ces ordinateurs présentent un inconvénient plus grand : le clavier à membrane. Les claviers à membrane ne procurent aucune sensation tactile, et l'espacement entre les touches est tel qu'il est impossible de taper très rapidement. La combinaison de ces inconvénients rend pénible l'entrée de programmes très longs. L'Oric-1, le Spectrum et le TO 7 tentent de résoudre ce problème en produisant un signal audible quand une touche a été suffisamment enfoncée. Mais cela ne constitue pas une compensation suffisante. De nombreuses sociétés proposent par exemple de meilleurs claviers — dimension normale et touches mécaniques — pour les ordinateurs Sinclair; mais les dispositifs bien conçus sont coûteux. Ils adoptent également la convention d'entrée de Sinclair qui permet de définir une commande BASIC à l'aide d'une seule touche.

La disposition idéale du clavier est celle où les rangées de touches vues latéralement forment un arc de cercle. Les mouvements des doigts de l'opérateur sont alors limités. Les seuls ordinateurs domestiques qui répondent à cette exigence sont le BBC, le Commodore 64 (ainsi que les derniers Vic-20), et l'Apple II.



**Le sténographe**
Quand il est nécessaire d'enregistrer un discours et que le sténographe n'a aucun moyen de ralentir l'orateur, un dispositif spécial de sténographie est utilisé. Des machines de ce type utilisent une sténographie phonétique. (Cl. Martin Burke.)

La disposition du clavier est depuis longtemps un sujet de mésentente pour les concepteurs. Lorsque les premières machines à écrire furent introduites au XIXe siècle, il y avait autant de dispositions de clavier que de fabricants; mais les touches les plus utilisées étaient généralement groupées au centre du clavier. Quand les premières machines à tiges d'impression furent introduites vers 1870, les fabricants découvrirent que même une frappe assez lente pouvait entraîner la collision de deux tiges. Ce problème survenait le plus souvent lors de la frappe de mots comme « tente », où les lettres les plus utilisées en français sont tapées dans une séquence très rapide. On décida de laisser sur le clavier un espace entre les lettres qui sont le plus souvent adjacentes dans des mots, d'où les dispositions QWERTY et AZERTY. Le clavier électronique adopta cette disposition pour suivre la norme existante. Voilà un exemple intéressant d'une norme, inadéquate aujourd'hui mais impossible à modifier sans provoquer de nombreuses difficultés.

Quelques tentatives ont cependant été faites pour développer d'autres claviers. En 1977, Mme Lillian G. Malt, profitant de la souplesse du matériel électronique, mit au point un clavier correspondant à la forme des mains. Il permettait une frappe beaucoup moins fatiguante. La frappe, nettement plus rapide, pouvait atteindre facilement trois cents mots par minute. Malheureusement, ce clavier ne réussit pas à briser la suprématie de la disposition AZERTY.

Une des caractéristiques intéressantes que ce clavier partageait avec ceux de nombreux micro-ordinateurs était sa possibilité de se détacher. La plupart des ordinateurs domestiques ne possèdent pas de moniteur intégré et sont assez petits pour être manipulés facilement. Tel n'est pas le cas pour plusieurs micro-ordinateurs à vocation professionnelle. Les claviers deviennent de plus en plus plats et sont reliés à l'ordinateur au moyen d'un câble. L'IBM PC Junior va un peu plus loin : la communication entre le clavier et le micro-ordinateur est similaire aux commandes à distance à infrarouge des téléviseurs et des magnétoscopes.

L'ergonomie n'étant pas une discipline entièrement objective (les relations entre les travailleurs et leur environnement subissent régulièrement des modifications), il n'est pas possible de donner des règles précises et immuables. L'objectif fondamental est le confort à long terme. Cette exigence entraîne un aménagement des outils qui permet à l'opérateur de se consacrer à la tâche à effectuer sans avoir à changer constamment de position et sans se fatiguer inutilement.

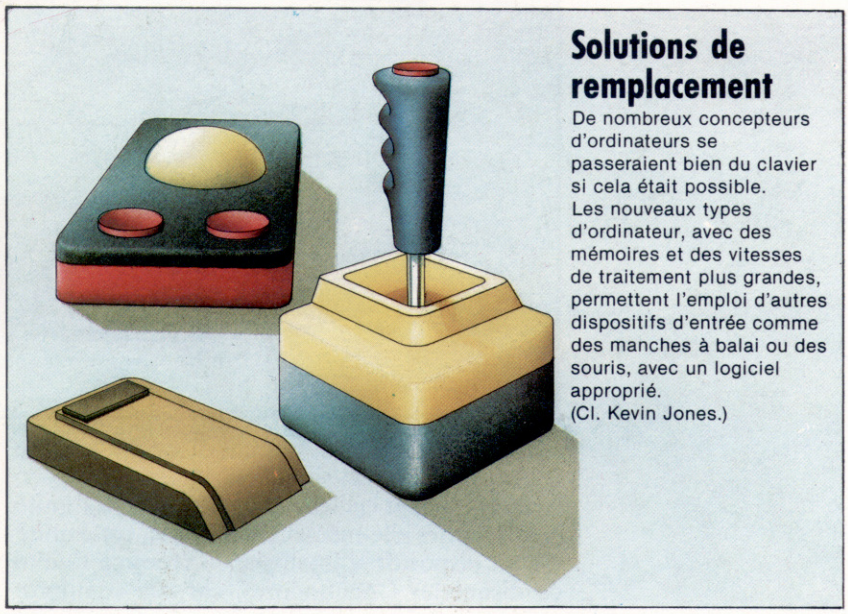
L'utilisateur d'ordinateur domestique peut tenter beaucoup de choses pour améliorer son environnement de travail. Quand nous avons parlé de l'Apple Lisa, nous avons souligné qu'il était possible de proposer des solutions de remplacement pour le clavier en travaillant avec un logiciel piloté par menu. Vous pouvez tenter une version peu coûteuse de cette nouvelle approche en utilisant une manette de jeu, et constater vous-même les avantages. Évidemment, vous devrez écrire de petits programmes d'application, mais, à l'aide d'instructions PEEK et POKE dirigées dans la mémoire écran, ce n'est pas très difficile.

**Frappe**
Avant l'arrivée des claviers électroniques, chaque touche devait être reliée physiquement au moulage du caractère responsable de l'impression sur le papier. D'où l'existence de contraintes au niveau de la disposition du clavier, puisqu'il fallait laisser une certaine distance entre les touches fréquemment utilisées pour éviter des collisions entre les tiges d'impression.
Ce n'est plus une obligation, mais...

Par ailleurs, si votre ordinateur vous permet de redéfinir la valeur des touches, vous pouvez redisposer le clavier, en collant des étiquettes qui indiquent leurs nouvelles valeurs. Dans ce cas, il est peut-être plus facile de lire (avec PEEK) la valeur des huit octets qui composent le caractère dans un tableau de huit variables, et de les écrire de nouveau (avec POKE). Vous pouvez écrire (POKE) les huit octets qui composent le caractère, directement dans l'espace affecté au caractère que vous désirez remplacer. Si vous utilisez cette méthode, n'oubliez pas de sauvegarder le premier ensemble de valeurs dans un tableau temporaire et de placer chaque caractère à sa nouvelle position. Sauvegardez sur cassette le programme qui effectue cette opération, car, si vous mettez l'ordinateur hors tension (ou à zéro), la valeur de chaque caractère reviendra à sa valeur initiale !

Finalement, si vous êtes bricoleur, vous pouvez construire un poste de travail sur mesure : le clavier repose sur une tablette distincte et le moniteur est incliné correctement. Les postes de travail proposés sur le marché permettent de placer le dispositif de stockage de masse (lecteurs de disquettes ou de cassettes) sur des étagères situées sous le plan de travail. L'ergonomie est, en fait, une simple question de bon sens. Quelques instants de réflexion peuvent aider à résoudre des problèmes comme le mal de dos ou la fatigue des yeux.

Il ne faudrait pas croire que le rôle de l'ergonomie est mineur dans la concurrence que se livrent les fabricants de matériels. Les récents succès des produits électroniques Olivetti au Japon, par exemple, montrent que la bonne technologie profite d'une ergonomie soignée.



## Solutions de remplacement

De nombreux concepteurs d'ordinateurs se passeraient bien du clavier si cela était possible. Les nouveaux types d'ordinateur, avec des mémoires et des vitesses de traitement plus grandes, permettent l'emploi d'autres dispositifs d'entrée comme des manches à balai ou des souris, avec un logiciel approprié.
(Cl. Kevin Jones.)

# Pistes, secteurs, blocs

**Le système d'exploitation de disquettes (DOS) doit se rappeler l'emplacement de tout ce qui est stocké. Sans un DOS, la programmation serait une tâche très difficile.**

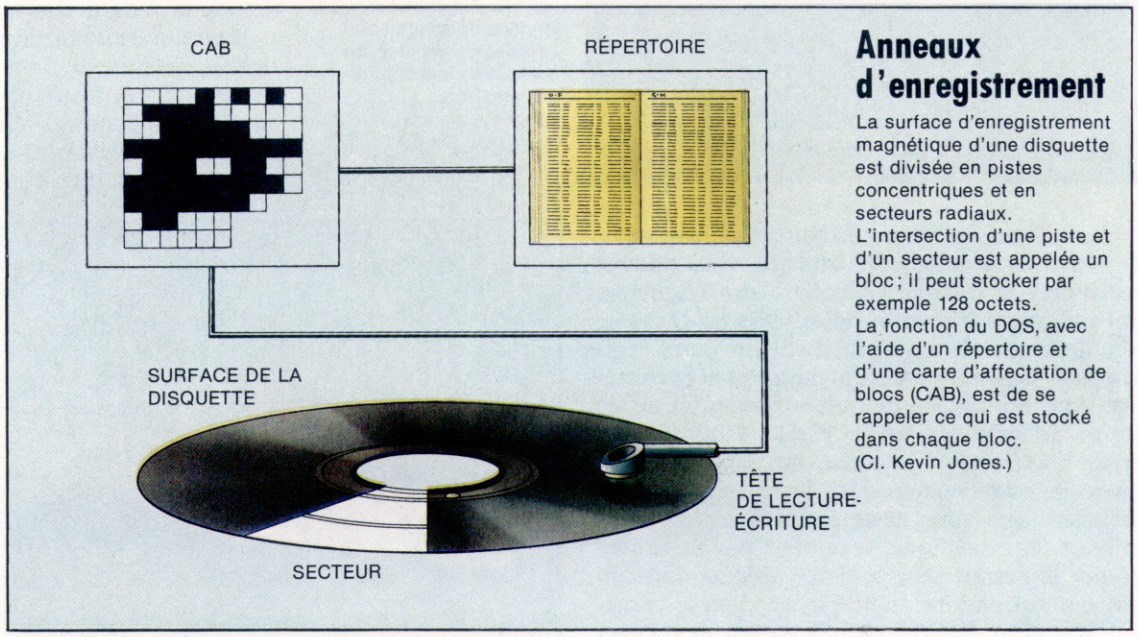
Avant qu'un ordinateur puisse être en mesure d'exécuter tout type de programme d'application, il doit d'abord disposer de son propre ensemble de programmes pour gérer les diverses parties du système et pour interpréter les instructions que renferme le programme de l'utilisateur. Sur la plupart des ordinateurs domestiques, le système d'exploitation réside en mémoire ROM. Généralement, nous ne pouvons absolument pas être conscients du fonctionnement du système d'exploitation. Ce type de fonctionnement est alors dit « transparent ».

Si votre système comporte un lecteur de disquettes, une large part de ce système d'exploitation sera destinée à la gestion des diverses opérations de disquettes. Nous nommons cet ensemble de routines le « système d'exploitation de disquettes », ou DOS *(Disk Operating System)*. Vous rencontrerez souvent ces trois lettres dans des noms de produit — le système d'exploitation de Microsoft, par exemple, se nomme MSDOS. Un DOS est normalement contenu sur trois types de supports. Il peut être logé en ROM à l'intérieur de l'ordinateur. Dans le Sinclair Spectrum, la ROM renferme les commandes d'exploitation du Microdrive intégré (qui n'est évidemment pas une disquette, mais le fonctionnement est similaire).

Le DOS peut également être logé dans l'unité de disquettes elle-même. Dans ce cas, cette unité est un dispositif « intelligent » (comme l'unité de disquettes Commodore), car elle comporte ses propres microprocesseurs, ROM et RAM. La construction de ces lecteurs est plus coûteuse, mais ces unités offrent des avantages considérables par rapport aux unités de disquettes « non intelligentes ». Par exemple, elles n'utilisent pas le précieux espace de la mémoire utilisateur, et peuvent se charger de façon autonome de tâches complexes de gestion de disquettes pendant que l'ordinateur continue à piloter le programme d'application.

Troisièmement, le DOS peut être logé à l'intérieur de la RAM de l'ordinateur. Cette technique devient de plus en plus populaire dans les systèmes de gestion où les lecteurs de disquettes sont intégrés dans l'ordinateur, et où la RAM est importante (la normale se situe au-delà de 128 K). Pour le fabricant, l'avantage est d'éliminer le besoin de créer un nouvel ensemble de ROM lors de chacune des modifications mineures qu'il désire apporter au DOS. De son côté, l'utilisateur bénéficie d'un large choix de systèmes d'exploitation proposés par divers constructeurs et qui fonctionnent sur son matériel sans difficultés.

Mais comment le DOS est-il chargé en mémoire RAM? Cette opération est effectuée dès la mise sous tension de l'ordinateur. Le DOS doit être transféré de la disquette à la RAM. Mais il n'existe aucun DOS dans l'ordinateur pour lui dire comment commander la disquette. Comment peut-il donc charger quelque chose en RAM? Un programme ne pouvant




**Anneaux d'enregistrement**

La surface d'enregistrement magnétique d'une disquette est divisée en pistes concentriques et en secteurs radiaux. L'intersection d'une piste et d'un secteur est appelée un bloc; il peut stocker par exemple 128 octets. La fonction du DOS, avec l'aide d'un répertoire et d'une carte d'affectation de blocs (CAB), est de se rappeler ce qui est stocké dans chaque bloc. (Cl. Kevin Jones.)

se charger par lui-même en RAM, l'ordinateur doit donc posséder un petit programme intégré en ROM qu'il exécute dès que la machine est mise sous tension. Ce programme se nomme « programme d'amorçage » et est en fait une forme de DOS très simple. La fonction d'amorçage consiste à trouver le DOS de la disquette et à le transférer octet par octet en RAM. A ce moment, le DOS peut entrer en action et commander des fonctions beaucoup plus complexes. Ce processus de mise sous tension de l'ordinateur et d'attente du chargement du DOS se nomme « amorçage ». Lorsqu'il est terminé, un message de sollicitation apparaît à l'écran pour indiquer que l'ordinateur est prêt à recevoir des commandes de l'utilisateur.

Quel que soit le support du système d'exploitation, sa principale fonction est de gérer le contenu de la disquette. Rappelons qu'une disquette est divisée en anneaux concentriques, les pistes, qui sont elles-mêmes divisées en secteurs; l'intersection d'une piste et d'un secteur est un bloc. Un bloc renferme généralement 128 octets d'information. Il représente la plus petite unité qu'une disquette puisse lire ou écrire. Une des principales fonctions d'un DOS est de permettre à l'ordinateur de se souvenir de l'emplacement exact de tout ce qui est stocké sur la disquette. Cette tâche est plus difficile qu'il ne paraît. Supposons que notre lecteur de disquettes dispose d'une capacité de 320 K — ce qui est assez pour stocker vingt programmes de 16 K chacun. Si chaque bloc renferme 128 K, pour charger l'un de ces programmes sans la présence d'un DOS vous devriez spécifier 128 blocs différents, en mentionnant pour chacun sa piste et son numéro de secteur!

## Répertoire local

| Nom de fichier | Type | Emplacement (piste-secteur) |
|---|---|---|
| PacMan | Prog | 20-1, 20-7, 20-2... |
| Température. | Prog | 25-11, 26-5, 26-12... |
| Budget | Prog | 23-12, 24-3, 24-9... |
| Budgetdon | Données | 27-1, 27-7, 27-2... |

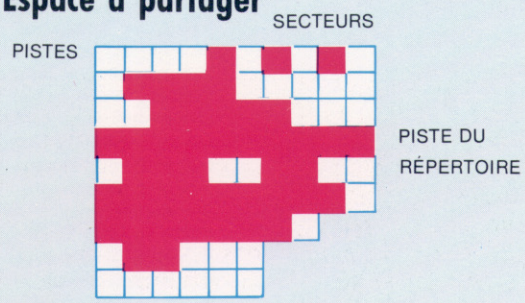
Le répertoire d'une disquette renferme la liste des noms et des types de fichier, et les numéros des pistes et des secteurs où le fichier est stocké.

Afin d'effectuer cette fonction, le DOS met continuellement à jour un répertoire de disquette. Ce répertoire est généralement logé en milieu de disquette pour réduire la distance que doit parcourir la tête de lecture-écriture. La vitesse de fonctionnement d'un lecteur de disquettes dépend beaucoup plus de la vitesse de déplacement de la tête d'une piste à l'autre que de la vitesse de rotation de la disquette.

Le répertoire est une liste de tous les fichiers (qui peuvent contenir des données ou des programmes) stockés sur une disquette. Cette liste mentionne le nom du fichier, son type et une liste des blocs (chacun étant désigné par la piste et le secteur) où le fichier est stocké. Le répertoire peut comprendre d'autres éléments, comme la date de la dernière mise à jour, ou une liste des utilisateurs qui peuvent utiliser un fichier particulier.

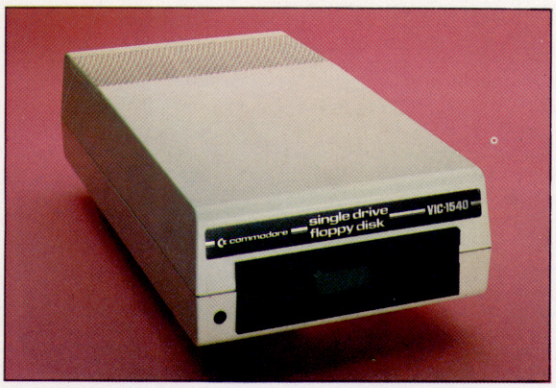
Lors du stockage d'un nouveau fichier, le DOS doit d'abord consulter la carte d'affectation de blocs (CAB). Cette partie de la mémoire est réservée pour stocker un ensemble de bits où chacun correspond à un bloc particulier sur la disquette; chaque fois qu'un bloc est utilisé, son bit passe de l'état zéro à l'état un. Certains ordinateurs domestiques munis de lecteurs de disquettes offrent un programme utilitaire qui affiche la carte d'affectation de blocs où vous pouvez surveiller les entrées qui sont faites lors de la sauvegarde d'un programme.

## Espace à partager




Avant que le DOS puisse stocker un nouveau fichier et faire une entrée dans le répertoire, il doit consulter la carte d'affectation de blocs (CAB) ou la liste des secteurs libres. C'est une section de la mémoire où chaque bit correspond à un bloc de la disquette. Le chiffre binaire 1 indique que le bloc est utilisé, et 0 qu'il est libre (nous avons représenté cela à l'aide de carrés pleins et vides).

Une autre caractéristique de ce système réside dans le fait que les fichiers ne sont pas stockés dans des blocs adjacents, comme on pourrait tout d'abord l'imaginer. Supposons qu'une piste soit composée de douze secteurs, numérotés de 1 à 12 dans le sens des aiguilles d'une montre. Les 128 premiers octets d'un programme pourraient être logés dans le secteur 1, les 128 octets suivants dans le secteur 2 et ainsi de suite.




**Test Q.I.**
Certains lecteurs de disquettes possèdent leurs propres microprocesseur et RAM. Ceux-ci sont appelés lecteurs « intelligents », et le DOS est intégré sous forme de ROM. Lorsque des lecteurs « non intelligents » sont utilisés, le DOS est stocké à l'intérieur de l'ordinateur.
(Cl. Chris Steven.)

Un système d'exploitation de disquettes comporte de nombreuses autres fonctions : formater de nouvelles disquettes (diviser une disquette vierge en pistes et secteurs et créer un répertoire vide), faire des copies de sauvegarde, « mettre de l'ordre » dans une disquette pleine.

# Toujours plus loin

**Le ZX-81 est toujours l'ordinateur le moins coûteux. Mais, avec des compléments adéquats, il peut devenir une machine très sophistiquée.**

Le Sinclair ZX-81 offre le meilleur rapport qualité-prix parmi les ordinateurs domestiques du marché, même dans sa version de base. Mais il existe un nombre impressionnant de dispositifs complémentaires qui le transforment en un système très évolué. Il peut créer des graphiques couleur haute résolution, offrir des fonctions de synthèse de la parole ou des possibilités de communication. Évidemment, l'ordinateur souffre de certaines limites. Mais elles peuvent être contournées par l'addition d'une variété d'articles déjà disponibles, comme des claviers professionnels, des modules de mémoire, etc.

**Module RAM**
En version standard, le ZX-81 ne possède que 1 K de RAM, et 123 octets sont occupés par des variables système. Par conséquent, l'extension de la mémoire est probablement le premier besoin de l'utilisateur. Les modules de mémoire Sinclair montrés sur cette photo ne sont disponibles que dans le format 16 K. (Cl. Marcus Wilson-Smith.)

**Également disponible**
En plus des unités représentées ici, il existe d'autres dispositifs visant à améliorer les performances du ZX-81. Par exemple, une carte couleur permet d'obtenir jusqu'à seize couleurs d'affichage, et un générateur de son peut produire trois « voix » programmables. Des ports bidirectionnels peuvent prendre en charge jusqu'à seize dispositifs d'entrée-sortie à la fois.
Le ZX-81 peut être étendu afin de lui permettre d'exploiter entièrement le potentiel de son microprocesseur Z80.

**Coupleurs acoustiques**
Il existe deux types de modem :
les modems connectés directement qui nécessitent une prise téléphonique supplémentaire, et les coupleurs acoustiques qui, comme ici, utilisent directement le combiné.
Les modems à connexion directe sont généralement plus chers. Ils génèrent et reconnaissent des signaux électroniques représentant les 0 et les 1 de l'information reçue ou transmise.
Les coupleurs acoustiques, pouvant être alimentés par piles, traduisent les 0 et les 1 en signaux sonores afin de les transmettre par le réseau téléphonique ; ils effectuent le processus inverse pour recevoir l'information.

**ROM Forth**
Les micro-ordinateurs Sinclair utilisent une version de BASIC plutôt particulière. Bien qu'il ne soit pas possible d'installer une autre version, l'utilisateur peut changer complètement de langage, en passant au FORTH par exemple. Il existe deux façons de le faire : soit en chargeant le nouveau langage en RAM à partir d'une unité à cassette (l'ordinateur revient au BASIC chaque fois qu'il est mis sous tension ou remis à zéro) ; ou en remplaçant la ROM BASIC par une autre. Le langage FORTH livré en ROM est très performant ; il permet d'exécuter plus de dix programmes simultanément. Cette possibilité peut être exploitée en programmation de commande lorsque plusieurs dispositifs doivent être programmés indépendamment.

**Synthèse de la parole**
Voici un autre dispositif complémentaire : l'unité de synthèse de la parole Sweet Talker de Cheetah. Sweet Talker, en raison de son mode de programmation, est beaucoup plus facile à programmer que la plupart des autres unités de synthèse de la parole. Il en existe d'autres semblables, pour le ZX-81 et pour de nombreux ordinateurs domestiques.

**Hebot**
La tortue Hebot, offerte en kit ou déjà montée, est un des plus sophistiqués des robots de sol. Il est livré avec un logiciel de commande complet, et il existe une vaste gamme de compléments tels que des photodétecteurs, qui peuvent être utilisés avec une bande réfléchissante collée au sol pour guider le robot selon une trajectoire prédéterminée.

**Claviers**
Le clavier à membrane du ZX-81 est sans doute son point faible. Il n'est donc pas surprenant que de nombreux constructeurs en proposent d'autres de dimensions standard et munis de touches mécaniques. Le clavier Mapsoft ZX-81, apparaissant sur la photo, est offert par Maplin Electronics, soit en kit, soit déjà monté.
En plus du jeu de caractères standard, le clavier Mapsoft offre trois touches supplémentaires. Pour moins de 400 F, il représente un dispositif complémentaire appréciable pour le ZX-81. Des produits similaires peuvent coûter deux fois plus cher. Il est également possible de se procurer un dessus de clavier de la même dimension que le clavier du ZX-81 et que l'on installe sur ce dernier. Il facilite la frappe mais son efficacité est limitée.

**Contrôleur de manette de jeu et manette de jeu**
De nombreux utilisateurs achètent le ZX-81 dans une perspective ludique. Il est donc étrange que Sinclair n'ait pas produit ses propres manettes de jeu et contrôleurs. Une grande variété de manettes de jeux est cependant offerte. Ces dispositifs peuvent être soit non programmables (l'unité spécifie pour vous quelles sont les touches activées par la manette), soit programmables (l'utilisateur peut alors définir quelles touches sont activées). Le modèle illustré ici, construit par AGF Hardware, peut être programmé en modifiant le montage électrique interne.

**Imprimante ZX**
L'imprimante Sinclair ZX utilise un papier traité sensible à l'électricité. Au lieu d'imprimer de façon conventionnelle, la tête d'impression enlève la couche d'aluminium et révèle ainsi un fond plus sombre. Alors que la vitesse d'impression est raisonnable, les restrictions au niveau du type de papier et de la largeur d'impression sont les principaux inconvénients de ce système.
Il est cependant possible d'utiliser une imprimante normale avec une carte interface. Des cartes sont offertes pour les interfaces RS232 et Centronics.

# Planification, efficacité

**En surveillant de près la structure des variables et du programme, vous pouvez accélérer l'exécution des programmes en basic.**

Malgré l'opinion de certains, le BASIC est un langage très souple et un puissant outil de formation. Vous pouvez écrire tout programme en BASIC pourvu que votre machine ait une mémoire suffisante et que le temps d'exécution demeure raisonnable. Cependant, puisque ce langage est interprété et non compilé, l'exécution des programmes peut être très lente, spécialement pour ceux qui exigent la traduction et l'exécution répétitive de la même instruction.

Le tri, par exemple, est un processus extrêmement répétitif : la procédure est effectuée à l'intérieur d'une boucle, et des petites boucles sont imbriquées à l'intérieur de la boucle principale. Si 100 éléments doivent être triés, le programme peut faire entre 2 500 et 5 000 itérations. Un tri BASIC sera toujours lent. Mais la façon dont le programme est écrit peut avoir une influence significative sur la vitesse d'exécution. Si une instruction doit être répétée 5 000 fois, et si le fait de soigner l'écriture du programme fait gagner 0,01 seconde à chaque itération, le gain est de 50 secondes, une amélioration considérable pour l'utilisateur.

Pour illustrer la différence entre une bonne et une mauvaise programmation, nous avons besoin d'un programme d'essai.

Le programme d'essai se présente ainsi :

```
1000 L = 500
2000 PRINT « *** DÉPART *** » : REM INSTRUCTIONS
SIGNAL SONORE ICI
2100 TI$ = « 000000 »
2200 FOR K = 1 10 L
-----------------------------------------------------
2900 NEXT K : T9 = TI
2950 REM INSTRUCTIONS SIGNAL SONORE ICI
3000 PRINT « *** STOP *** »
3100 PRINT « CELA A MIS »; (T9/60); « SECONDES »
```

Les lignes 2100 et 3100 s'adressent aux utilisateurs Commodore. Pour les autres machines, effacez-les et remplacez-les par les instructions appropriées. L'espace entre les lignes 2200 et 2900 est l'endroit où nous insérerons le code à chronométrer. Notez que tous les chronométrages référeront à L répétitions où L est la limite de la boucle. Une vérification d'une seule exécution d'une partie de programme serait très imprécise puisque l'horloge ne fonctionne qu'en soixantièmes de seconde et qu'il y a aussi un intervalle de synchronisation imposé par le programme d'essai.

Voici par ordre d'importance quelques règles générales pour écrire efficacement en BASIC :

### 1. Éviter toutes opérations arithmétiques dans les boucles.

Les fonctions exponentielles ($x^3$, signifiant x élevé à la puissance 3) et les fonctions trigonométriques (cos x, signifiant le « cosinus de l'angle » x) sont particulièrement lentes. La multiplication et la division sont plus lentes que l'addition et la soustraction. Mais même la plus rapide de ces opérations est relativement lente.

Insérez ces lignes dans le programme d'essai :

```
900 Z = 1.1
2300 X = Z↑3
```

et exécutez le programme. Sur notre machine d'essai, 500 répétitions prirent 27,95 secondes. Remplacez maintenant la ligne 2300 par :

```
2300 X = Z * Z * Z
```

et exécutez le programme. Cela a pris 3,55 secondes, une différence considérable!

Une étude plus approfondie révélera l'exposant de la puissance où il devient préférable de remplacer la répétition des multiplications par la fonction exponentielle. Sur notre ordinateur, ce seuil se situait à la puissance 18 (X = Z ↑ 18). Rappelons cependant que pour calculer $Z^{2,3}$, par exemple, la multiplication répétée serait inadéquate alors que la fonction exponentielle fonctionne pour tous les nombres réels, y compris les nombres négatifs.

Utilisez le programme d'essai pour constater le temps requis pour effectuer d'autres opérations arithmétiques, et essayez de trouver les meilleures solutions. Est-il plus rapide de diviser un nombre par 2 ou de le multiplier par 0,5?

### 2. Utiliser des variables plutôt que des constantes numériques.

Chaque fois qu'une constante numérique (7280 par exemple) est employée dans une instruction BASIC, ce nombre doit être traduit sous une forme utilisable. Essayez cette ligne :

```
2300 X = X + 7280
```

Sur notre machine 500 répétitions mirent 4,63 secondes, alors que :

```
900 C = 7280
2300 X = X + C
```

exécuta 500 répétitions en 2,75 secondes.

### 3. Si vous devez utiliser l'instruction GOTO, sautez vers l'avant de votre programme. Si cependant vous devez effectuer un branchement vers l'arrière, passez au début du

programme plutôt que de passer quelques lignes en arrière.

La même règle s'applique pour GOSUB. En rencontrant une instruction GOTO ou GOSUB, l'interpréteur BASIC compare le numéro de ligne cible avec le numéro de ligne en cours. Si le numéro cible est plus grand que le numéro en cours, l'interpréteur cherche vers l'avant, ligne par ligne, jusqu'à ce que la ligne cible soit trouvée. Mais, si la cible est moins grande que la ligne en cours, la recherche commence au tout début du programme. Il est donc plus efficace de placer les sous-programmes et les sections fréquemment utilisés aux deux extrémités du programme. Ajoutez 56 lignes REM au début du programme pour simuler une longueur typique, et essayez :

```
2300 GOTO 2400
2400 GOTO 2500
2500 GOTO 2900
```

500 répétitions furent exécutées en 2,33 secondes, alors que :

```
2300 GOTO 2500
2400 GOTO 2900
2500 GOTO 2400
```

mit 4,85 secondes.

4. Initialiser toutes les variables dans l'ordre de la fréquence d'accès.

Les noms de variables sont stockés par l'interpréteur dans un tableau, dans l'ordre de leur première apparition dans le programme. La dernière variable mentionnée aura le temps d'accès le plus long. Pour la même raison, vous devriez éviter d'utiliser une nouvelle variable dans un programme lorsque vous pouvez avoir recours à une variable qui a été déjà utilisée mais qui n'est pas employée à ce moment.

Si une variable est utilisée à l'intérieur de boucles imbriquées — c'est fréquent dans les programmes de tri — il est préférable de l'initialiser au début du programme avant toute autre variable, avec une valeur factice si nécessaire.

```
1000 L = 500 : C = 7280 : X = 1.1
2300 A = 0
```

prit 2,2 secondes pour 500 répétitions, alors que :

```
1000 A = 0 : L = 500 : C = 7280 : X = 1.1
2300 A = 0
```

mit 2.06 secondes.

5. Éviter d'utiliser des chaînes.

Les opérations de chaînes n'utilisent pas la mémoire comme les opérations arithmétiques. Un programme système, nommé « programme récupérateur », doit être appelé fréquemment par l'interpréteur pour mettre de l'ordre dans le contenu de la mémoire de chaîne. Cette procédure prend beaucoup de temps.

Une démonstration générale est difficile à écrire parce que la gestion de la mémoire varie beaucoup d'un ordinateur à l'autre : vous devez remplir de données une grande partie de la mémoire utilisateur — un grand tableau numérique fera l'affaire — puis effectuer des manipulations de chaîne qui nécessiteront l'appel du programme récupérateur. Sur notre machine nous entrons :

```
40 POKE 52, 32 : POKE 56, 32 : CLR
```

pour réduire sérieusement l'espace mémoire disponible pour les programmes BASIC, puis nous entrons :

```
1000 L = 500 : DIM T$(L)
1100 FOR K = 1 TO L
1200 T$(K) = « A » + « B »
1300 PRINT K
1400 NEXT K
```

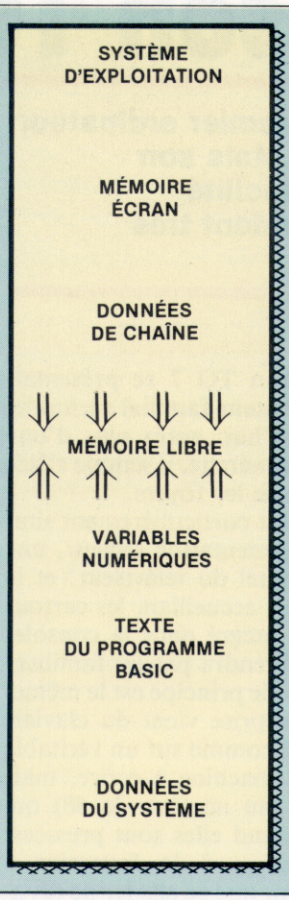
ce qui utilise beaucoup de mémoire de chaîne et met en place un tableau de chaîne pour une utilisation ultérieure. L'instruction PRINT est exécutée lors de chaque itération, affichant la valeur du compteur de boucle. Lorsque nous avons exécuté cette version du programme d'essai, une pause survenait entre chaque impression, car le programme récupérateur était appelé pour remanier la mémoire. Parfois, la pause dura plus de 3 secondes. Le programme continue :

```
2300 A$ = LEFT$(T$(L), 1) : B$ = A$ + RIGHT$(T$(L), 1)
```

500 répétitions furent exécutées en 30,03 secondes. Lorsque nous avons exécuté le même programme avec beaucoup plus de mémoire disponible, la récupération des positions inutilisées n'était plus visible et la même boucle fut exécutée en 8,66 secondes.

## Topographie mémoire

Voici la topographie mémoire simplifiée d'un ordinateur domestique typique. La plupart des microprocesseurs peuvent adresser jusqu'à 64 K (65 536 octets) qui sont répartis en RAM, ROM et espace inutilisé. Pour ce qui est de la vitesse d'exécution d'un programme BASIC, un des facteurs les plus importants est le mode de stockage des chaînes. Lorsque le contenu d'une chaîne est modifié, une copie entièrement nouvelle est réalisée en mémoire. La totalité de l'espace mémoire disponible est éventuellement utilisée et BASIC appelle le récupérateur afin de mettre en ordre la mémoire de chaînes.




Ensemble des programmes standard, contenus en ROM, nécessaires pour le fonctionnement interne.

Chaque octet de cette portion de la RAM correspond à une position de caractère sur l'écran.

Lorsqu'une chaîne est définie ou modifiée, les caractères sont stockés dans cette section de la RAM.

La mémoire libre diminue quand la liste de variables ou la longueur des chaînes augmentent.

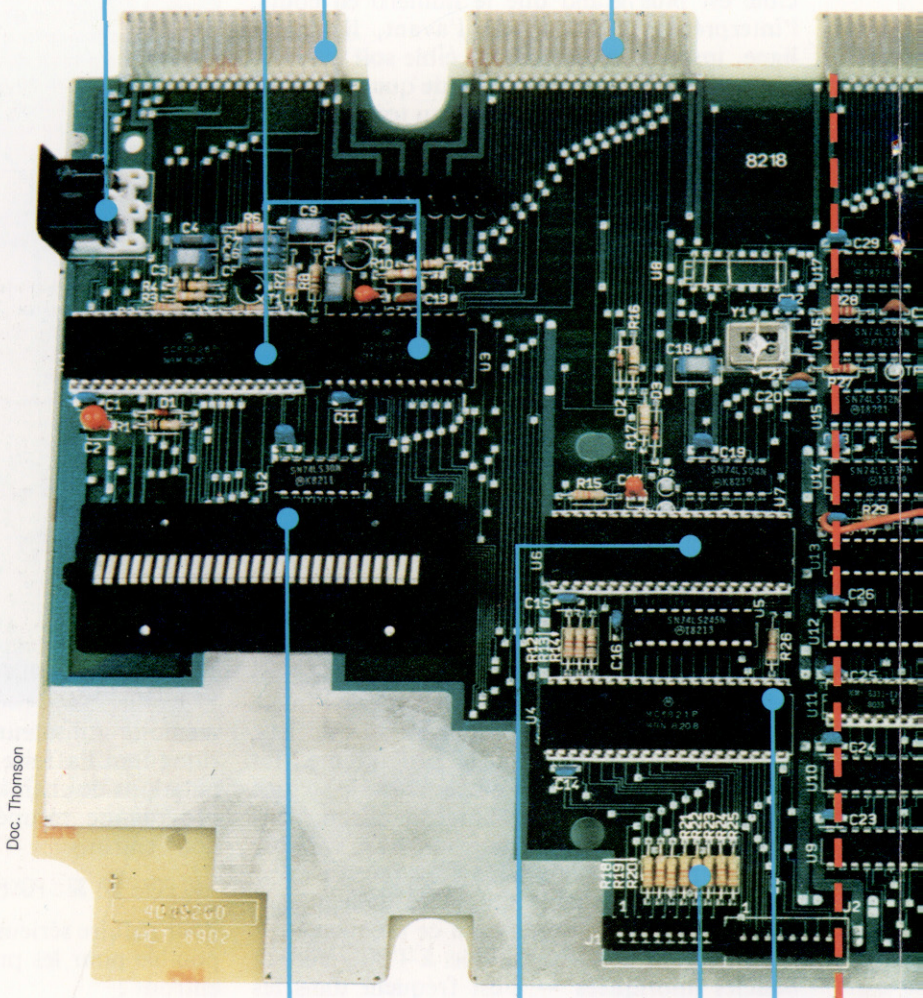
Les variables numériques occupent généralement sept octets chacune.

Le texte d'un programme est stocké ici, généralement sous la forme de codes ASCII.

Tous les ordinateurs utilisent une partie de leur RAM pour stocker des variables internes et des tampons pour le clavier ou pour l'unité à cassette. (Cl. Liz Dixon.)

# Thomson TO 7

**Un peu cher, le premier ordinateur familial français. Mais son graphisme et sa facilité d'utilisation le rendent très attrayant.**




Doc. Thomson

A sa sortie, le Thomson TO 7 se présentait comme le premier ordinateur familial véritablement français. Aujourd'hui, après plus d'une année de présence sur le marché, il semble s'être fait une bonne place dans les foyers.

Sa mise en fonction est particulièrement simple : une fiche pour l'alimentation secteur, une autre pour la prise Péritel du téléviseur, et le tour est joué. Le boîtier, accueillant les cartouches programmes, est intégré dans la console. Son maniement ne surprendra pas les familiers des lecteurs de cassettes, le principe est le même. La première (petite) surprise vient du clavier. Pas de touches en relief comme sur un véritable ordinateur ou sur une machine à écrire, mais des touches sensitives (au nombre de 58) qui émettent un « bip » quand elles sont pressées. Malgré sa présentation « à la française » (AZERTY), il est évident que ce clavier ne favorise pas les rois de la frappe rapide. Il ne faut donc pas perdre patience lorsqu'on veut entrer un ensemble d'instructions — en BASIC bien sûr — dans le TO 7

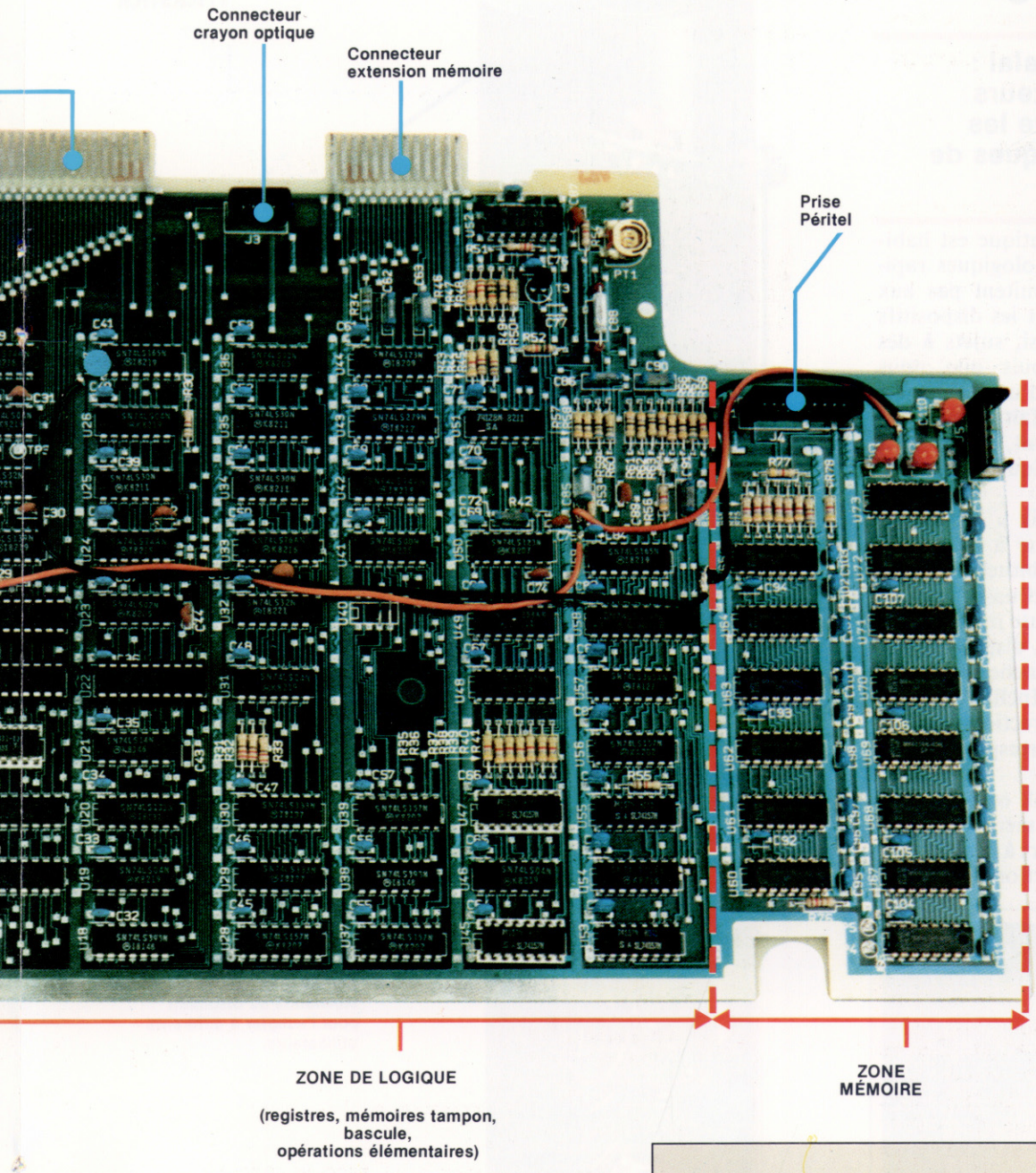
Une deuxième surprise, plus importante que la précédente et plus agréable : le crayon optique qui trouve une « niche » au-dessus du clavier. Grâce à lui et au logiciel adapté, vous pouvez dessiner en couleurs sur l'écran. Seul inconvénient, celui de rester près du récepteur de télévision et donc de se fatiguer les yeux. Cela dit, l'image est de bonne qualité. Elle est stable et sa définition suffisante (8 couleurs; affichage de 25 lignes de 40 caractères avec minuscules accentuées; résolution de 320 × 200 points). Après avoir placé une cartouche dans le lecteur Mémo 7 situé à gauche du clavier, mis sous tension le système et placé l'interrupteur en position marche, l'écran se colore en bleu et le TO 7 vous propose trois chemins possibles : un réglage du crayon optique; une utilisation de la cartouche logiciel choisie; l'exploitation du clavier.

Pour les débutants ou les plus jeunes, le crayon optique peut être, à ce stade, très utile dans la mesure où il permet d'entrer directement des commandes dans l'ordinateur en appuyant simplement les endroits désirés sur l'écran. C'est une bonne approche pour se familiariser avec l'appareil sans avoir à se heurter immédiatement à la manipulation du clavier.

Le Thomson TO 7 s'est beaucoup enrichi en logiciel depuis quelques mois. L'alliance du constructeur avec VI-FI Nathan a surtout favorisé les logiciels de jeux éducatifs, mais des éditeurs indépendants donnent de plus en plus au TO 7 des dimensions variées.

Il faut noter que les caractères du TO 7, sur matrice 8 × 8, sont compatibles vidéotex. Cela signifie qu'avec l'interface que doit sortir Thomson le TO 7 se transformera selon vos désirs en terminal (intelligent) dans les réseaux Télétel qui se développent actuellement en France.

A moins que vous ne préfériez attendre la sortie — imminente — du premier frère du TO 7...




## THOMSON TO 7

**PRIX**
***

**DIMENSIONS**
270 × 450 × 80 mm.

**POIDS**
3,5 kg.

**HORLOGE**
1 MHZ.

**MÉMOIRE**
6 K ROM résident et 16 K max. sur Mémo 7; 22 K RAM (14 K pour la gestion de l'écran et 8 K pour l'utilisateur).

**AFFICHAGE VIDÉO**
8 couleurs, 200 lignes de 320 points. Mémoire graphique directement accessible (crayon optique intégré).

**INTERFACES-EXTENSION**
Module de communication, contrôleur de musique et de jeux, lecteur-enregistreur de programmes et de fichiers, lecteur et contrôleur de disquette, imprimante.

**LANGAGE INTÉGRÉ**
BASIC.

**AUTRES LANGAGES OFFERTS**
LOGO TO 7; ASSEMBLEUR.

**CLAVIER**
A membranes.

**DOCUMENTATION**
Assez claire mais relativement peu technique.

Doc. Thomson

# Tubes...

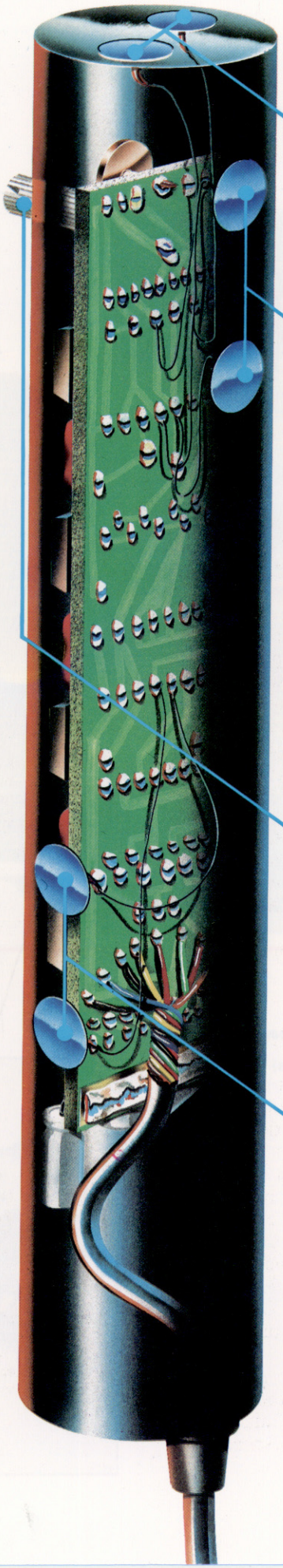
## Nouveaux manches à balai : l'un utilise des interrupteurs au mercure, l'autre capte les signaux électromagnétiques de votre corps.

L'industrie de la micro-informatique est habituée aux développements technologiques rapides. Les changements ne se limitent pas aux ordinateurs; les périphériques et les dispositifs complémentaires sont, eux aussi, sujets à des perfectionnements. Ainsi, depuis que nous avons présenté le mécanisme d'un manche à balai, deux nouveaux types ont fait leur apparition sur le marché. Ils ont, en fait, entièrement abandonné le système mécanique que nous avions décrit précédemment.

Un dispositif nommé le « Stik » fut le premier manche à balai analogique à rejeter les mécanismes de signalisation habituels. Le Stik consiste en une poignée munie d'un bouton de mise à feu et d'un bouton de pause monté sur le côté. Contrairement aux autres dispositifs qui sont montés sur un socle, ce manche à balai est simplement tenu dans la main. Incliné par rapport à la verticale dans la direction désirée, il provoque un mouvement correspondant de l'image associée sur l'écran.

Le mécanisme de base du Stik est formé de quatre tubes scellés remplis de mercure. Lorsque le tube est incliné par rapport à la verticale, le mercure coule dans la direction choisie et

**Commande directe**
Le fonctionnement du Trickstick est fondé sur le « bruit du secteur », qui est la radiation électromagnétique produite par le conducteur de bouclage de chaque maison. Votre corps agit comme une antenne émettant des bruits que le Trickstick capte.



**Trickstick**

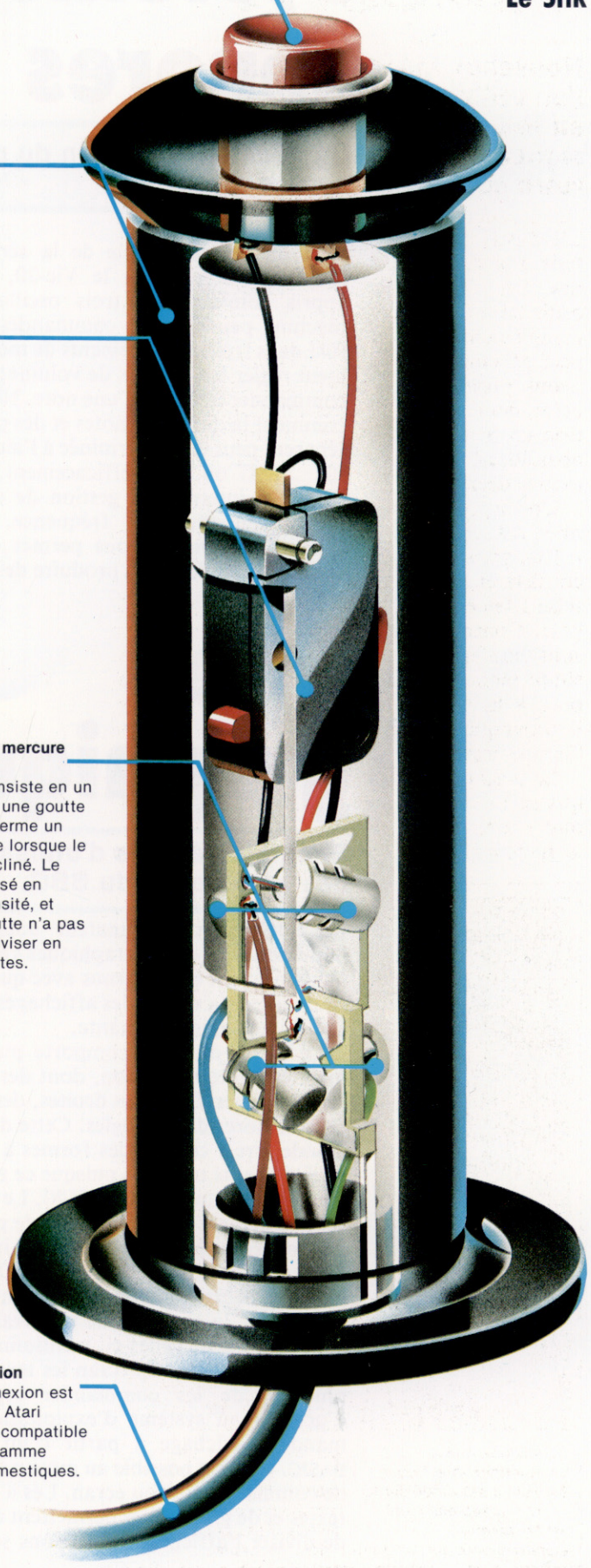
**Commandes de mouvements horizontaux**
En plaçant le pouce sur une de ces deux touches à effleurement, l'utilisateur commande les déplacements avant et arrière.

**Commandes de mouvements verticaux**
La touche du haut commande le mouvement vers le haut, celle du bas commande le mouvement vers le bas.

**Commande de sensibilité**
Permet d'ajuster la sensibilité du dispositif pour l'adapter à différents utilisateurs.

**Boutons de mise à feu**
Chacun de ces boutons génère un signal indépendant : vous pouvez lâcher des bombes avec le bouton du bas et déclencher, simultanément, un canon laser avec l'autre.

## Le Stik




effectue un ou plusieurs contacts électriques, comme si un interrupteur avait été fermé. Remettre la poignée dans une position verticale permet au mercure de revenir dans le tube, ce qui coupe le contact. La réponse du système est vraiment meilleure que celle des manches à balai précédents.

La toute dernière méthode pour convertir des mouvements manuels en signaux pouvant être compris par un ordinateur est utilisée par le Trickstick. Ce manche à balai est unique en raison de l'effet électrique qu'il emploie : il se sert du corps humain comme antenne pour capter le bruit du secteur (la radiation électromagnétique inoffensive émise par le conducteur de bouclage dans toute pièce). Le Trickstik est un tube de plastique scellé tenu verticalement à deux mains. Trois paires de touches sensitives sont placées sur la surface du tube; une des paires commande le déplacement avant-arrière; une autre commande les déplacements vers le haut et vers le bas; la dernière correspond aux boutons de mise à feu.

Le bruit de secteur capté par le corps humain est transmis par ces touches vers le circuit sensitif. Les impulsions, converties en signaux, fournissent alors à l'ordinateur une information directionnelle. Les signaux peuvent aussi exprimer une distance de mouvement. Le niveau du signal et donc la rapidité de la sortie sont proportionnels à la pression appliquée sur la touche. De cette manière, le Trickstick combine la commande graduelle du manche à balai analogique et la commande numérique rapide et directe d'une unité à interrupteur. Comme l'effet de transmission électromagnétique varie d'une personne à l'autre, le Trickstick doit être ajusté pour chaque utilisateur. Un bouton monté à l'une des extrémités du tube permet cette opération.

Ce dispositif est certainement révolutionnaire. Les fabricants ont déposé une demande de brevet d'invention pour cette technique, mais la fiabilité et les performances du dispositif restent à prouver.

# Systèmes sonores

**Un deuxième examen du potentiel sonore du Vic-20.**

Lors du dernier article de la série « Son et lumière » concernant le Vic-20, nous avons appris comment les trois oscillateurs de la machine peuvent être commandés en écrivant POKE dans trois emplacements de mémoire; comment régler les niveaux de volume; et comment commander la durée d'une note. Nous avons vu comment la durée des notes et des pauses qui les séparent peut être déterminée à l'aide de boucles FOR ... NEXT ou, plus efficacement, à l'aide de l'horloge interne. La gestion de ces trois éléments musicaux — fréquence, volume et synchronisation — vous permet de créer des mélodies simples et de produire des effets sonores intéressants.

## Composer des mélodies

Pour construire une mélodie, vous devez d'abord rassembler les notes nécessaires. Voici les notes que nous pourrions utiliser pour créer une mélodie :

*ré # mi fa ré # do sol # sol sol # ré # ré*

A l'aide des techniques décrites précédemment, la durée des notes et des pauses peut être définie en utilisant la variable TI. Notre mélodie peut donc être jouée en exécutant le programme suivant (notez l'utilisation de variables pour simplifier la sélection des instructions POKE) :

```
10 V = 36878
20 FOR I = 1 TO 11
30 READ N : REM *NOTE*
40 POKE V, 7 : P = TI : REM *VOLUME ON*
50 IF TI - P < 15 THEN 50 : REM *PAUSE*
60 POKE V - 3, N : D = TI : REM *JOUER NOTE*
70 IF TI - D < 20 THEN 70 : REM *DURÉE*
80 POKE V - 3, 0 : REM *ARRÊT NOTE*
90 NEXT I
100 DATA 203, 207, 209, 203 : REM *VALEURS DES NOTES*
110 DATA 195, 187, 179, 175
120 DATA 179, 203, 203
130 POKE V, 0 : REM *VOLUME OFF*
140 END
```

# Effet graphique

**Premières étapes d'utilisation des graphiques du BBC.**

Le BBC est un ordinateur domestique très populaire. Des effets graphiques vraiment étonnants peuvent être obtenus avec quelques lignes en BASIC, et la vitesse des affichages produits est également impressionnante.

Le BASIC du BBC comporte plusieurs commandes haute résolution, dont des instructions pour dessiner des lignes droites, des points, tracer et colorer des triangles. Cette dernière commande sert à colorer des formes à l'aide d'une série de petits triangles puisque ce BASIC n'offre pas de commande du type PAINT. Le BBC ne possède également pas de commande pour dessiner des cercles et des ellipses, et il est impossible de programmer des plans-objets. Cependant, il offre plusieurs fonctions intéressantes et inhabituelles que la majorité de ses rivaux ne possèdent pas. Parmi celles-ci, mentionnons la possibilité de mélanger à l'écran les textes et les graphiques, de les commander séparément et d'accéder au système d'exploitation qui commande l'affichage à partir d'un programme BASIC. Cela est possible au moyen de l'ensemble de commandes VDU ou écran. Les « fenêtres » de textes et de graphiques permettent à l'utilisateur de diviser l'affichage en sections séparées pour les graphiques et pour le texte. Différentes couleurs peuvent être définies pour chaque fenêtre et chacune peut être effacée indépendamment.

## Mode d'affichage

Le BBC possède huit modes texte, dont trois se chargent uniquement des affichages de texte. Vous pouvez choisir des affichages d'une largeur de 20, 40 ou 80 caractères, selon le mode retenu. Deux, quatre ou seize couleurs sont disponibles, de nouveau selon le mode retenu, mais à l'intérieur de ces modes de couleurs limitées, le programme peut choisir deux ou quatre couleurs parmi les seize disponibles dans le mode couleur supérieur.

Le mode 7 est différent de tous les autres puisque le jeu standard de caractères ASCII et les codes associés ne sont pas utilisés. L'affichage est composé de caractères spéciaux. Les commandes graphiques normales, comme PLOT et DRAW, ne fonctionnent pas dans le mode 7.

Le tableau suivant illustre les choix de résolution et de couleur offerts par la sélection des différents modes :

| Mode | Texte | Graphiques | Couleurs |
|---|---|---|---|
| 0 | 80 × 32 | 640 × 256 | 2 |
| 1 | 40 × 32 | 320 × 256 | 4 |
| 2 | 20 × 32 | 160 × 256 | 16 |
| 3 | 80 × 25 | | 2 (noir et blanc) |
| 4 | 40 × 32 | 320 × 256 | 2 |
| 5 | 20 × 32 | 160 × 256 | 4 |
| 6 | 40 × 25 | | 2 (noir et blanc) |
| 7 | 40 × 25 | | Télétexte |

Ce programme ne fait que jouer les notes dans la bonne séquence avec des durées et des pauses égales. Par conséquent, la mélodie n'est pas parfaite. Avec de l'expérience, vous pouvez construire des programmes plus complexes.

## Effets sonores

En utilisant deux des trois oscillateurs, il est possible de créer des harmonies simples. Le programme donné ci-dessous joue l'harmonie du *ré* majeur (*fa, la* et *ré*) en commençant par le *fa* et en ajoutant le *la* et le *ré* en insérant un écart d'une seconde entre chaque son. L'harmonie continue alors pendant deux autres secondes.

```
10 POKE 36878, 7
20 POKE 36874, 233 : D = TI
30 IF TI - D < 60 THEN 30
40 POKE 36875, 219 : D = TI
50 IF TI - D < 60 THEN 50
60 POKE 36875, 147 : D = TI
70 IF TI - D < 120 THEN 70
80 POKE 36878, 0 : POKE 36874, 0
90 POKE 36875, 0 : POKE 36876, 0
100 END
```

Beaucoup peut cependant être fait pour rendre ces sons plus intéressants. Par exemple, le volume peut être varié pendant la durée d'une note; on le fait alors monter ou descendre en fonction d'une variable. Par exemple :

```
100 V = 36878
110 FOR I = 1 TO 12
120 POKE V, I
130 NEXT I
140 POKE V, 0
```

Le volume monte ainsi par pas de 1 jusqu'à une valeur maximale de 12, alors que la gamme complète de volume va de 0 (nul) à 15 (fort). Le volume peut aussi varier rapidement en utilisant alternativement des réglages de volume élevés et faibles. De même, la fréquence peut varier pour provoquer le fléchissement d'une note en changeant la ligne 120 en :

```
POKE V - 3, 203 + I
```

Il est également intéressant d'essayer différentes combinaisons de bruit, de fréquences d'oscillateurs et de volumes. Il est ainsi possible de produire des sons plus agréables. Que ce soit pour créer de la musique ou pour produire des effets sonores, le but est de réduire les répétitions de notes monotones.

Nous avons démontré comment les fonctions sonores simples du Vic-20 pouvaient être utilisées pour produire d'intéressantes séquences de notes. Le principal problème est le manque de commandes sonores, ce qui implique l'utilisation d'instructions BASIC complexes pour effectuer des tâches relativement simples. De longues routines empêchent l'interpréteur BASIC de traiter assez rapidement le code compris entre chaque note. La seule manière d'éviter ce problème est de se procurer l'un des nombreux logiciels commerciaux qui offrent des commandes supplémentaires pour la programmation musicale. La cartouche Commodore Super Expander offre une gamme complète de commandes sonores, ainsi qu'une fonction servant à stocker les mélodies écrites à l'aide de la cartouche. Mais si vous désirez utiliser une machine offrant des fonctions musicales et sonores évoluées, dirigez-vous vers d'autres modèles.

L'origine de l'écran haute résolution est définie dans le coin inférieur gauche de l'écran, quel que soit le mode choisi. Les valeurs verticales vont de 0 à 1023, et les valeurs horizontales de 0 à 1279. Cette méthode constante de topographie de l'écran est très pratique lorsque vous décidez de modifier l'affichage en passant d'un mode à un autre. De plus, si le mode d'affichage est changé pendant l'exécution d'un programme, l'écran est automatiquement effacé.

Les couleurs d'arrière-plan, de texte et des graphiques sont définies à l'aide des commandes COLOUR et GCOL. Le BBC utilise la notion intéressante de couleurs logiques et réelles pour permettre à l'utilisateur de sélectionner un ensemble limité de couleurs parmi les seize couleurs offertes. Prenons un exemple en utilisant le mode 0 où deux couleurs uniquement peuvent être spécifiées. Les deux couleurs possibles de premier plan reçoivent les numéros de couleurs logiques 0 et 1 et, sauf autre commande donnée à l'ordinateur, celui-ci interprète 0 comme noir et 1 comme blanc. La commande COLOUR sélectionne la couleur de premier plan du texte. COLOUR 1 sélectionnerait la couleur logique 1 comme couleur de texte, mais il est possible de redéfinir cette couleur à l'aide de l'une des commandes VDU. VDU19 définit la couleur logique. Pour définir la couleur logique 1 comme vert (dont le numéro de couleur réelle est 2), la commande suivante doit être envoyée :

```
VDU19, 1, 2, 0, 0, 0,
```

Les trois zéros n'ont aucune signification et sont destinés à une future extension du système.

La commande GCOL est associée à deux nombres. Le second nombre est le numéro de la couleur logique pour un affichage graphique; le premier nombre définit la façon dont la couleur est utilisée à l'écran. Pour la commande GCOL a, b, les valeurs de a peuvent aller de 0 à 4. Cela permet à l'utilisateur de spécifier si le point ou la ligne doit être affiché dans la couleur de premier plan, si les opérateurs ET, OU ou OU exclusif doivent définir la couleur par rapport à la couleur déjà présente, ou si la couleur originale doit être inversée.

Dans un prochain article « Son et lumière », nous reviendrons au BBC et expliquerons les fonctions graphiques haute résolution, la définition de caractères, et examinerons de plus près l'ensemble des commandes VDU.

# Récupération

**Après avoir examiné comment insérer de nouveaux enregistrements, nous passons aux manières de les extraire. Comme prévu, nous rencontrons le problème de trouver une correspondance exacte.**

Nous avons terminé le dernier article par un exercice où vous deviez écrire un programme de base de données qui accepte l'entrée de données. Examinons certaines des étapes impliquées dans l'entrée d'un nouvel enregistrement afin de continuer notre examen des éléments impliqués dans la partie INITIALISATION de notre programme principal. D'abord, supposons que nous avons les champs suivants et leurs tableaux correspondants :

| CHAMPS | TABLEAU |
|---|---|
| 1 champ NOM | NOMCHP$ |
| 2 champ NOM MODIFIÉ | MODCHP$ |
| 3 champ RUE | RUECHP$ |
| 4 champ CODE POSTAL | CPOSCHP$ |
| 5 champ VILLE | VILLECHP$ |
| 6 champ TÉLÉPHONE | TELCHP$ |
| 7 champ INDEX | NDXCHP$ |

La signification de ces champs doit être suffisamment claire, à l'exception, éventuellement, des champs 2 et 7. Examinons d'abord le champ NOM MODIFIÉ. Lorsque nous nous sommes initialement penchés sur le problème du format de données pour le nom, nous nous sommes demandé s'il devait être strictement spécifié ou libre et nous avons opté pour la dernière solution. Puisque la forme d'entrée d'un nom peut être extrêmement variable, un format rigide aurait rendu l'écriture de recherche et de tri très difficile. Pour résoudre ce problème, nous avons décidé de convertir tous les noms dans un format standard : toutes les lettres converties en majuscules, et tous les caractères non alphabétiques (comme les espaces, les points, les apostrophes, etc.) supprimés en ne laissant qu'un seul espace entre le nom de famille (s'il y a lieu) et le prénom.

Il est nécessaire de standardiser les noms parce que les routines de tri et de recherche doivent être en mesure de comparer ce qui est comparable. D'autre part, lorsque nous extrayons un nom et une adresse dans la base de données, nous désirons qu'ils soient présentés tels qu'ils ont été entrés initialement. On peut traiter ce problème de deux manières : soit chaque nom fiché est converti en une forme standard uniquement lors des tris et des recherches, soit le champ du nom peut être converti en une forme standard et stocké dans un fichier distinct afin que les routines de tri et de recherche puissent avoir un accès immédiat aux noms standardisés.

Les deux approches ont des avantages et des inconvénients. La conversion temporaire des champs de noms, lorsqu'ils sont sollicités par d'autres routines, économise l'espace mémoire, puisque moins de données sont stockées dans le fichier. En revanche, cette procédure prend beaucoup de temps.

L'autre champ que nous devons examiner de près est le champ INDEX. Il est désigné comme un champ supplémentaire pour permettre l'extension ou la modification futures de la base de données sans avoir à réécrire le programme. Son utilisation introduit la notion d'« édition de liens », une expression qui signifie l'établissement de relations entre les données et les unités de traitement. Tous les champs ou éléments de chacun des enregistrements sont reliés parce qu'ils ont tous le même index (le même numéro d'élément, ou indice, dans leurs tableaux respectifs), et parce que tous les champs d'un enregistrement seront stockés ensemble dans un fichier. Cela rend l'addition de nouveaux types de données difficile et implique éventuellement une complète réorganisation de la structure du fichier et la nécessité de réécrire une grande partie du programme. L'insertion du champ INDEX à ce stade rendra tout changement ultérieur au programme beaucoup plus simple.

Lorsqu'un nouvel enregistrement est ajouté, il est plus simple de le faire à la fin du fichier (c'est-à-dire au premier élément vide de chaque tableau). Il est presque certain que le nouvel enregistrement ne sera pas ordonné par rapport aux autres. Voilà un problème que nous pourrons envisager plus tard. La première chose à faire est de trouver la taille du tableau. Puisque c'est un élément d'information qui risque d'être utile dans plusieurs parties du programme, il est préférable de le placer dans INITIALISATION. Voilà un cas où nous aurons évidemment besoin d'une variable globale (c'est-à-dire une variable qui peut être utilisée dans toute partie du programme). Nous l'appellerons TAILLE. L'indice de l'enregistrement en cours est une autre variable globale qui sera probablement utile. Puisque aucun enregistrement ne sera en cours lors du début d'exécution du programme, l'affectation d'une valeur initiale à ACT n'aura lieu que lorsque le programme traitera les données. ACT peut cependant être initialisé à 0 dans la procédure INITIALISATION. L'initialisation d'une variable n'est pas strictement nécessaire en BASIC puisqu'elle se fait automatiquement. C'est cependant une bonne habitude à prendre, et qui devient indispensable dans le cas des variables locales afin d'éviter des interférences lors de l'utilisation de la même variable ailleurs. Lors de l'exécution initiale du programme, divers

types d'initialisation surviennent, et les données sont chargées à partir d'une disquette ou d'une cassette et transférées dans des variables de chaînes. Le menu CHOIX est alors présenté. Si l'utilisateur choisit l'option 6 (ajouter un enregistrement au fichier), la valeur retournée de la variable CHOIX est 6, qui appelle le sous-programme AJOUTENR. AJOUTENR suppose qu'une valeur a déjà été affectée à TAILLE et qu'il peut commencer à solliciter des entrées (remarque : cela suppose également qu'INITIALISATION a déjà correctement déclaré les tableaux nécessaires).

Ajouter un nouvel enregistrement signifie aussi que le fichier peut déjà n'être plus en ordre. Puisqu'un tri peut prendre du temps, il peut ne pas être nécessaire de trier les enregistrements après chaque addition (une décision que nous pouvons repousser pour l'instant). Nous définirons, à la place, un drapeau pour indiquer qu'un nouvel enregistrement a été ajouté.

Nous sommes maintenant en mesure de composer une liste des tableaux, variables et drapeaux qui pourront être requis par le programme.

**TABLEAUX**

NOMCHP$ (champ du nom)
MODCHP$ (champ du nom modifié)
RUECHP$ (champ de la rue)
CPOSCHP$ (champ du code postal)
VILLECHP$ (champ de la ville)
TELCHP$ (champ du numéro de téléphone)
NDXCHP$ (champ d'indice)

**VARIABLES**

TAILLE (taille actuelle du fichier)
ACT (indice du fichier en cours)

**DRAPEAUX**

AJ (nouvel enregistrement ajouté)
TRI (trié depuis la dernière modification d'enregistrement)
SAV (sauvegarde effectuée depuis la dernière modification d'enregistrement)
RMOD (modification apportée depuis la dernière sauvegarde)

Il est probable que, au cours du développement du programme, quelques autres tableaux seront nécessaires. Pour les drapeaux, il est évident que, même si d'autres seront nécessaires, les quatre mentionnés ci-dessus peuvent ne pas être tous requis. Une sauvegarde ou un tri ne sera nécessaire (en supposant que le fichier soit déjà trié) que si une modification est apportée. RMOD est probablement le seul drapeau vraiment nécessaire. Mais, si nous décidons d'utiliser les quatre drapeaux, le sous-programme INITIALISATION devrait leur affecter les valeurs appropriées. Avec cet exercice de programmation descendante, voyons comment il est facile d'écrire *AJOUTENR*.

**I 4 (EXÉCUTION) 6 (AJOUTENR)**

```
DÉBUT
  Déterminer la taille actuelle du fichier
  Solliciter des entrées
  Affecter les entrées aux extrémités des tableaux
  Définir le drapeau RMOD
FIN
```

**II 4 (EXÉCUTION) 6 (AJOUTENR)**

```
DÉBUT
  (la taille du fichier est TAILLE)
  (sollicitation d'entrées)
  Effacer l'écran
  Afficher premier message de sollicitation pour le premier
    tableau (TAILLE)
  Entrer les données dans le tableau (TAILLE)
  (solliciter et entrer tous les tableaux)
  Définir RMOD à 1
Fin
```

Tout cela est direct et n'implique aucune boucle ou autres structures complexes. Il est donc possible d'écrire directement le programme en BASIC. Signalons que TAILLE est une variable définie pendant l'exécution d'INITIALISATION et qu'elle n'a pas à être codée dans cette section.

**III 4 (EXÉCUTION) 6 (AJOUTENR) CODE BASIC**

```
CLS : REM OU UTILISER PRINT CHR$(24) POUR EFFACER
  L'ÉCRAN
INPUT « ENTREZ LE NOM »; NOMCHP$(TAILLE)
INPUT « ENTREZ LA RUE »; RUECHP$(TAILLE)
INPUT « ENTREZ LE CODE POSTAL »; CPOSCHP$(TAILLE)
INPUT « ENTREZ LA VILLE »; VILLECHP$(TAILLE)
INPUT « ENTREZ LE NUMÉRO DE TÉLÉPHONE »;
  TELCHP$(TAILLE)
LET RMOD = 1
LET NDXCHP$ = STR$(TAILLE)
GOSUB *MODNOM*
RETURN
```

La troisième ligne avant la fin définit le champ NDXCHP$ égal à la valeur de TAILLE (convertie en une chaîne par STR$), afin qu'il puisse servir d'index ultérieurement. Le sous-programme *MODNOM*, appelé juste avant la fin du programme, n'est rien d'autre que le programme décrit en détail précédemment. Quelques petites modifications devront être apportées. Ce sous-programme sert à recevoir l'entrée libre du nom et à la convertir dans une forme standard. La sortie de ce sous-programme sera un élément (TAILLE) dans un tableau nommé MODCHP$. Toutes les recherches et tous les tris peuvent maintenant être conduits sur les éléments de MODCHP$, puisque l'élément aura le même indice que les autres champs de l'enregistrement. Il sera facile d'afficher le nom et l'adresse tels qu'ils furent entrés initialement. En d'autres termes, la recherche sera faite sur MODCHP$, mais l'affichage proviendra de NOMCHP$.

C'est à peu près tout ce qui est impliqué dans l'addition d'un nouvel enregistrement au fichier, bien que nous n'ayons prévu ni vérification d'erreur ni intervention dans l'éventualité où il n'y aurait plus de place dans le tableau. Puisque tous nos programmes sont écrits sous la forme de modules, de telles modifications et améliorations peuvent être effectuées ultérieurement, sans avoir à écrire de nouveau le programme au complet.

Les sous-programmes MODENR et EFFENR (servant respectivement à modifier et à effacer des enregistrements) sont similaires à AJOUTENR, sauf qu'avant de les exécuter, nous devons localiser

l'enregistrement que nous désirons modifier. Par conséquent, ces deux sous-programmes commenceront par appeler TROUVENR. Ce sous-programme est fondé sur une routine de recherche que nous avons déjà rencontrée. La principale différence est qu'il est certain, cette fois, qu'il ne peut y avoir deux éléments de données identiques, puisqu'il est très rare que deux personnes aient des noms parfaitement identiques.

Il existe deux façons de mener une recherche. La première consiste à travailler sur une pile non ordonnée. La recherche est alors plus lente qu'elle ne devrait être. Dans le pire des cas, le programme pourrait avoir à passer en revue tous les éléments de données avant de localiser l'élément recherché.

Dans le second cas, les données sont ordonnées de quelque façon, soit numériquement, soit alphabétiquement. Le programme n'aura alors qu'à effectuer la recherche sur une petite portion des éléments de la liste.

Une autre façon d'accélérer la routine de tri serait de maintenir une table de référence des emplacements dans le tableau de la première apparition de chaque lettre de l'alphabet. Cette table devrait cependant être mise à jour soigneusement lors de tout changement apporté aux données.

Voici un programme en pseudo-langage qui recherche les éléments dans le tableau MODCHP$. La variable de chaîne CLÉ$ est l'élément essentiel de la recherche. Le terme « clé » signifie ici le groupe de caractères utilisé pour spécifier quel enregistrement est nécessaire.

```
Sollicitation du nom à rechercher
CLÉ$ = nom à rechercher
BAS = 1
RECHERCHE = 0
HAUT = TAILLE
BOUCLE pendant que (BAS < = HAUT) ET (RECHERCHE = 0)
  MID + INT ((BAS + HAUT)/2)
  SI CLÉ$ = MODCHP$(MID)
    ALORS
      AFFICHER NOMCHP$
      AFFICHER RUECHP$
      AFFICHER CPOSCHP$
      AFFICHER VILLECHP$
      AFFICHER TELCHP$
      RECHERCHE = 1
    SINON
      SI CLÉ$ > MODCHP$(MID)
        ALORS BAS = MID + 1
        SINON HAUT = MID - 1
      FINSI
    FINSI
FINBOUCLE
SI RECHERCHE = 0 ALORS AFFICHER « ENREGISTREMENT
  NON TROUVÉ »
FIN
```

Cette section de pseudo-langage s'inspire du programme déjà utilisé pour rechercher des pointages de football, mais vous verrez qu'elle produit une sortie adéquate si l'enregistrement ne peut être trouvé (la dernière instruction PRINT). Elle ne sera exécutée que si la boucle ne réussit pas à localiser une correspondance exacte entre CLÉ$ et MODCHP$(MID).

Malheureusement, une correspondance exacte est peu probable, même si le nom et le numéro de téléphone que vous recherchez sont dans la base de données. C'est parce que l'instruction SI CLÉ$ = MODCHP$ n'offre aucune souplesse; elle ne permet pas la moindre différence entre la chaîne de caractères entrée par l'utilisateur en réponse à une sollicitation et la chaîne de caractères stockée dans MODCHP$(MID). Dans un carnet d'adresses ordinaire, l'œil parcourt la page et est en mesure de reconnaître un nom même s'il n'est pas parfaitement écrit. L'ordinateur en est incapable.

Il existe cependant des manières de contourner ce problème, bien que cela implique un effort supplémentaire de programmation et allonge le temps d'exécution. La première amélioration serait de ne vérifier que le nom de famille; il devient donc inutile de stocker le nom dans MODCHP$ sous la forme NOM DE FAMILLE (espace) PRÉNOM. Nous avons déjà développé une routine pour inverser l'ordre d'un nom dans notre cours de programmation BASIC (voir « Variantes de basic »); cette méthode peut être intégrée comme une routine lors de la création du champ MODCHP$.

Après avoir localisé avec succès la première apparition du nom de famille recherché, la routine TROUVENR doit alors vérifier si la partie prénom de cet élément est identique à l'entrée du nom (CLÉ$). Si c'est le cas, il n'y a aucun problème — l'enregistrement a été localisé. Sinon, les choses commencent à se compliquer, et nous devons soigneusement planifier notre stratégie. Nous pourrions par exemple parcourir tous les prénoms, et, si une correspondance exacte n'est pas trouvée, commencer à rechercher une correspondance approximative. Voici la difficulté : qu'est-ce qui constitue une correspondance approximative?

Au lieu d'afficher le message ENREGISTREMENT NON TROUVÉ, il serait peut-être préférable de produire un message du genre CORRESPONDANCE EXACTE NON TROUVÉE, ESSAYER UNE CORRESPONDANCE APPROXIMATIVE? (O/N)?. Que signifient les mots « correspondance approximative »? Rob est-il une correspondance approximative de Robert? Et Robrt? Tous les deux représentent des entrées possibles dans le programme TROUVENR. Essayons de définir ce que nous voulons dire par correspondance approximative et commençons à développer un programme en BASIC pour trouver la correspondance qui se rapproche le plus de la chaîne entrée.

Supposons que la chaîne en mémoire est ROBERT. Laquelle de ces deux chaînes représente la correspondance la plus proche : ROB ou RBRT? La seconde a quatre lettres exactes parmi six, alors que la première n'en a que trois. Par contre, la première donne les trois lettres dans le bon ordre, alors que la seconde n'en donne que deux.

Le choix est ici arbitraire. Nous décidons de donner la priorité à une correspondance exacte

entre CLÉ$ et une sous-chaîne du nom en mémoire. S'il n'y a aucune correspondance exacte avec une sous-chaîne, le programme essaiera d'obtenir le maximum de lettres communes. Voici le programme formulé en termes d'entrée et de sortie :

**ENTRÉE**

Une chaîne de caractères.

**SORTIE**

La correspondance qui se rapproche le plus de cette chaîne de caractères.

Le programme suivant, écrit dans un pseudo-langage proche du BASIC, effectuera une recherche dans toutes les chaînes d'un tableau et examinera les *n* premières lettres où *n* est le nombre de lettres dans CLÉ$. S'il n'y a aucune correspondance, un message l'annoncera :

```
DIM TABLEAU$(4)
FOR L = 1 A 4
LIRE TABLEAU$(L)
NEXT L
DATA « ROBERT », « RICHARD », « ROBIANA », « ROBERTA »
CLÉ$ = « RON »
LCLÉ = LEN(CLÉ$)
RECHERCHE = 0
BOUCLE FOR INDEX = 1 A 4
  SI CLÉ$ = LEFT$(TABLEAU$(INDEX), LCLÉ)
    ALORS PRINT « CORRESPONDANCE EST »;
TABLEAU$(INDEX)
      RECHERCHE = INDEX
  FINSI
FINBOUCLE
SI RECHERCHE = 0
  ALORS PRINT CLÉ$; « N'A PAS DE CORRESPONDANCE
EXACTE »
  PRINT « DANS LES »; LKEY; « PREMIERS CARACTÈRES »
```

Le programme pourrait alors rechercher des groupes de caractères d'une longueur LKEY, en commençant par le second caractère de chaque chaîne. Si aucune correspondance n'est trouvée, il pourrait rechercher dans les groupes commençant au troisième caractère, et ainsi de suite. Finalement, s'il n'existe aucune correspondance de trois caractères dans les chaînes, le programme pourrait essayer de trouver quelle chaîne a le plus grand nombre de lettres en commun avec CLÉ$.

Nous pourrions en fait écrire des pages sur ces techniques de recherche approximative employées dans les logiciels commerciaux. La plupart offrent la possibilité d'effectuer une recherche sur les tout premiers caractères du champ, comme le programme que nous venons de développer. D'autres peuvent extraire un enregistrement si la séquence spécifiée de caractères apparaît quelque part dans le champ, ou ailleurs dans l'enregistrement. La possibilité d'utiliser des caractères génériques est aussi particulièrement utile; ainsi, spécifier M?R permettrait de trouver MARIE, MIREILLE, mais pas MAUD. La forme de recherche la plus sophistiquée permet d'établir des correspondances phonétiques; ainsi, entrer POULAIN trouverait aussi POULIN.

## Variantes de basic

**SPECTRUM**

Listage du programme servant à inverser l'ordre du nom de famille et du prénom donné précédemment :

```
100 CLS
200 PRINT « ENTREZ LE NOM SOUS LA
FORME »
300 PRINT « PRÉNOM NOM DE FAMILLE »
400 PRINT « EX : JEAN DUPONT »
500 INPUT « ENTREZ LE NOM »; N$
600 GOSUB 9500
700 PRINT « LE NOM SOUS FORME
STANDARD EST »
800 PRINT N$
1000 STOP
9500 REM S - P POUR INVERSER L'ORDRE
DU NOM
9520 GOSUB 9600
9540 IF P = 0 THEN RETURN
9560 LET N$ = F$ + « , » + P$
9580 RETURN
9600 REM S - P POUR SÉPARER LE NOM
SUR L'ESPACE
9620 LET N = LEN(N$)
9630 LET P = 0
9640 FOR K = 1 TO N
9650 IF FN M$(N$, K, 1) = « » THEN LET P =
K : LET K = N
9660 NEXT K
9670 IF P = 0 THEN RETURN
9680 LET P$ = FN L$(N$, P - 1)
9700 LET F$ = FN (N$, N - P)
9720 RETURN
9900 REM FONCTIONS UTILISATEUR
9990 DEF FN M$ (X$, P, N) = X$ (P TO P + N - 1)
9991 DEF FN L$ (X$, N) = X$ ( TO N)
9992 DEF FN R$ (X$, N) = X$
      (LEN X$ - N + 1 TO )
```

**LEFT$**

Sur le Commodore 64, sur le Vic-20, sur l'Oric-1 et sur le Lynx, remplacez les lignes 9600 à 9720 du listage du Spectrum par ces lignes :

```
9600 REM S - P POUR SÉPARER N$ SUR
L'ESPACE
9620 LET N = LEN (N$)
9630 LET P = 0
9640 FOR K = 1 TO N
9650 IF MID$ (N$, K, 1) = « » THEN LET P =
K : LET K = N
9660 NEXT K
9670 IF P = 0 THEN RETURN
9680 LET P$ = LEFT$ (N$, P - 1)
9700 LET F$ = RIGHT$ (N$, N - P)
9720 RETURN
```

**RIGHT$**

**MID$**

et effacez les lignes 9900 à 9992.

**INSTR**

Sur le Dragon 32 et sur le BBC, remplacez les lignes 9600 à 9720 du listage du Spectrum par ces lignes :

```
9600 REM S - P POUR SÉPARER N$ SUR
L'ESPACE
9620 LET N = LEN (N$)
9640 LET P = INSTR (N$, « »)
9670 IF P = 0 THEN RETURN
9680 LET P$ = LEFT$ (N$, P - 1)
9700 LET F$ = RIGHT$ (N$, N - P)
9720 RETURN
```

et effacez les lignes 9900 à 9992.

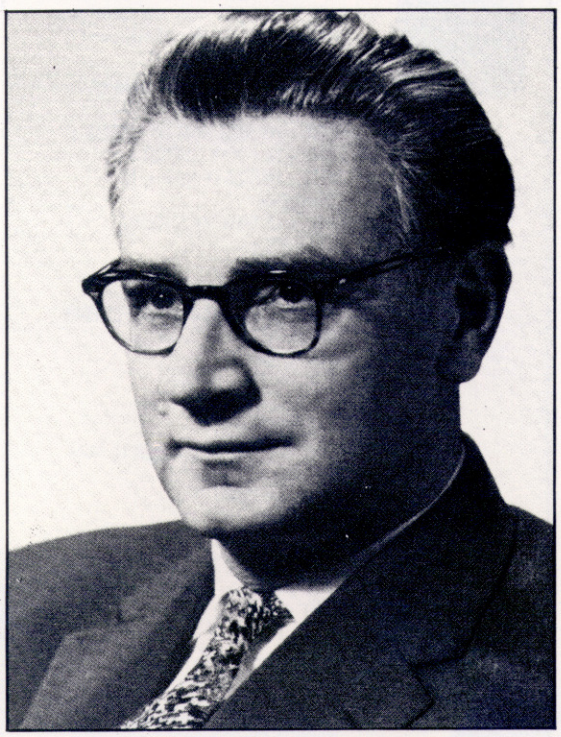
Comme nous l'avons déjà mentionné, INSTR est une fonction très utile, particulièrement dans des applications de base de données.

**INPUT**

Sur le BBC, remplacez la ligne 500 du listage du Spectrum par :

```
500 INPUT « ENTREZ LE NOM », N$
```

# Konrad Zuse



Siemens Museum, Munich

## Mêmes recherches au même moment : Neumann aux États-Unis et Zuse en Allemagne.

Des inventions sont souvent faites simultanément dans différentes parties du monde à partir de notions qui ont été développées indépendamment. En 1940, alors que le premier ordinateur à lampes (ENIAC) était développé aux États-Unis, un ingénieur allemand, Konrad Zuse, travaillait à la conception d'un calculateur programmable.

Zuse est né à Berlin le 22 juin 1910. Après des études à l'université technique de Berlin, il travailla comme ingénieur aéronautique pour la société d'aviation Henschel, où il concevait des ailes d'avion. Les principes mathématiques de base dans ce genre de recherche furent énoncés au cours des années vingt. Mais les calculs individuels requis pour la production de chaque paire d'ailes exigeaient la contribution d'équipes complètes travaillant avec des machines à additionner mécaniques et avec des règles à calcul. Zuse perçut vite le besoin d'une machine qui pourrait effectuer ce travail rapidement. Le soir, avec des amis, dans l'appartement de ses parents, il commença à envisager de construire un ordinateur qui pourrait effectuer cette tâche.

Sa première machine, le Z1, était un dispositif mécanique qui pouvait effectuer les quatre opérations arithmétiques élémentaires, calculer des racines carrées, et convertir des nombres décimaux en notation binaire et vice versa. Bien que ne connaissant pas les travaux de Charles Babbage, dont la machine de différence avait été créée pour effectuer des calculs laborieux, il arriva à plusieurs conclusions similaires et à certaines qui allaient beaucoup plus loin. Le principal apport de Zuse fut de reconnaître qu'un interrupteur pouvait, par ses deux états, être utilisé comme moyen de stockage de données ou comme dispositif de commande.

Zuse continua à essayer de représenter les données et les instructions sous forme binaire, et en 1941 il commença à construire un ordinateur électromagnétique qu'il nomma le Z2. En plein effort de guerre, le gouvernement allemand ne manifesta que peu d'intérêt pour son invention. Cependant, le potentiel militaire de celle-ci fut rapidement reconnu et des fonds furent alloués à Zuse pour développer le nouveau Z3. Il s'agissait d'un ordinateur électrique, avec câblage électrique au lieu des liaisons mécaniques utilisées dans les machines précédentes, qui permit une conception plus compacte et plus élégante.

Zuse construisit le Z3 malgré des handicaps sérieux. Les bombardements des Alliés l'obligèrent à déménager son atelier plusieurs fois. La pénurie de matériel, due à la guerre, l'obligea à récupérer divers composants téléphoniques et à utiliser de vieilles pellicules cinématographiques qu'il perfora pour représenter des codes de huit trous par colonne, au lieu des rubans de papier.

Le Z3 pouvait stocker 64 mots d'une longueur de 22 bits. L'information était entrée au moyen d'un clavier et les résultats étaient affichés à l'aide d'un arrangement de lampes montées sur un panneau. Malheureusement, le Z3 fut détruit ainsi que tous les autres ordinateurs de Zuse lors des bombardements intensifs de Berlin.

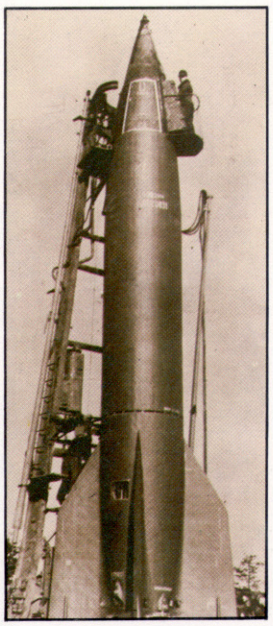
L'un de ses ordinateurs fut adapté par la société d'aviation Henschel pour accélérer la construction de la bombe volante HS-293. Il s'agissait d'un avion sans pilote qui était lancé d'un bombardier et guidé vers sa cible par commande radio.

Le dernier ordinateur construit par Zuse pendant la guerre, le Z4, eut la longueur de ses mots portée à 32 bits. Il fut évacué vers Göttingen lorsque les Alliés approchaient de Berlin. Il fut finalement transporté à Bâle, en Suisse, où il fonctionna jusqu'en 1954.

Zuse fut incapable de fabriquer des ordinateurs dans l'Allemagne d'après guerre et se concentra sur des recherches théoriques. Il développa un langage sophistiqué nommé PLANKALKÜL qui pouvait traiter logiquement des problèmes mathématiques ou des informations plus générales.

Lorsqu'il fut de nouveau en mesure de construire des ordinateurs, il forma la société Zuse, qui fut le plus important constructeur d'ordinateurs en Allemagne jusqu'en 1969, où cette société fut intégrée dans le groupe Siemens.

**Bombe volante**
Les ordinateurs de Zuse furent développés pour remplacer des équipes de techniciens qui travaillaient avec des règles à calcul pour effectuer divers calculs aéronautiques. Ils furent surtout appliqués à la conception des V1 et V2 (voir photo), les bombes volantes utilisées pendant la Seconde Guerre mondiale.
(Cl. Imperial War Museum, Londres.)



# PROGRAMME N° 7

## CARACTÈRES

### TABLEAU ASCII

| Numéro de code | Caractère ASCII | Numéro de code | Caractère ASCII |
|---|---|---|---|
| 32 | espace | 62 | > |
| 33 | ! | 63 | ? |
| 34 | " | 64 | ⓐ |
| 35 | # | 65 | A |
| 36 | $ | 66 | B |
| 37 | % | 67 | C |
| 38 | & | 68 | D |
| 39 | ' | 69 | E |
| 40 | ( | 70 | F |
| 41 | ) | 71 | G |
| 42 | * | 72 | H |
| 43 | + | 73 | I |
| 44 | , | 74 | J |
| 45 | – | 75 | K |
| 46 | . | 76 | L |
| 47 | / | 77 | M |
| 48 | 0 | 78 | N |
| 49 | 1 | 79 | O |
| 50 | 2 | 80 | P |
| 51 | 3 | 81 | Q |
| 52 | 4 | 82 | R |
| 53 | 5 | 83 | S |
| 54 | 6 | 84 | T |
| 55 | 7 | 85 | U |
| 56 | 8 | 86 | V |
| 57 | 9 | 87 | W |
| 58 | : | 88 | X |
| 59 | ; | 89 | Y |
| 60 | < | 90 | Z |
| 61 | = | | |

### THÈME

Après avoir abordé les nombres aléatoires et les parties entières et utilisé les fonctions RND et INT, il s'agit cette fois de jongler avec les lettres. Comment? Grâce à l'instruction CHR$, qui traduit des nombres en lettres. La machine se réfère au code ASCII, dans lequel chaque caractère correspond à un nombre défini.
Par exemple :

```
? CHR$ (65)    A
? CHR$ (82)    R
```

On peut utiliser CHR$ avec INT et RND pour générer les lettres de manière aléatoire.
Ainsi CHR$ (64 + INT (RND (1)*26 + 1)).
Cette instruction opère un tirage aléatoire de lettre.
Voici maintenant un petit exemple amusant d'utilisation des instructions CHR$, RND et INT. Ce petit programme est en même temps une synthèse de l'utilisation des commandes déjà étudiées.

Vous êtes un cambrioleur. Vous recherchez le code secret permettant d'ouvrir un coffre-fort richement garni. Vous devez donc trouver le code secret que connaît votre ordinateur. Entrez une lettre de votre choix et la machine vous indiquera si elle se trouve placée dans l'alphabet avant ou après le bon code. Vous disposez de cinq essais avant que la police ne vous arrête.

```
§LIST

5   HOME
10  INVERSE : HTAB 14 : PRINT
   "LE CAMBRIOLEUR" : NORMAL
12  VTAB 5: PRINT
   "VOUS ETES UN CAMBRIOLEUR."
13  VTAB 6: PRINT
   "VOUS RECHERCHEZ UN CODE SECRET"
14  VTAB 7: PRINT
   "PERMETTANT D'OUVRIR UN COFFRE-FORT"
15  VTAB 8: PRINT
   "BIEN GARNI."
16  VTAB 9: PRINT
   "VOUS DEVEZ TROUVEZ LE CODE SECRET QUE"
17  VTAB 10: PRINT
   "CONNAIT VOTRE ORDINATEUR."
18  VTAB 12: PRINT
   "ENTREZ UNE LETTRE DE VOTRE CHOIX ET IL"
19  VTAB 13: PRINT
   "VOUS INDIQUERA SI ELLE SE TROUVE PLACEE"
20  VTAB 14: PRINT
   "DANS L'ALPHABET AVANT OU APRES LE BON"
21  VTAB 15: PRINT "CODE"
22  VTAB 17: PRINT
   "VOUS DISPOSEZ DE 5 ESSAIS AVANT QUE LA"
23  VTAB 18: PRINT
   "POLICE NE VOUS ARRETE"
30  VTAB 22: INVERSE :
   INPUT "TAPEZ RETURN POUR CONTINUER";
    RE$: NORMAL

80  LET C$ =
    CHR$ (64 + INT (RND (1) * 26 + 1))
85  HOME
90  FOR I = 1 TO 5
92  PRINT : PRINT
97  PRINT "ESSAIS NO :" I
98  PRINT : PRINT : INPUT G$
110 IF G$ = C$ THEN GOTO 210
120 IF G$ < C$ THEN PRINT
    "LE BON CODE EST PLUS"
130 IF G$ > C$ THEN PRINT
    "LE BON CODE EST MOINS"
140 PRINT : PRINT "LOIN QUE :"; G$
150 NEXT I
160 PRINT
170 PRINT "CLIC CLAC VIVE LES MENOTTES"
175 PRINT
180 PRINT "VOUS ETES FAIT"
185 PRINT : PRINT :
    PRINT "LA POLICE VOUS ARRETE"
186 PRINT
190 INVERSE : PRINT
   "LE BON CODE ETAIT"; C$: NORMAL
200 PRINT : INPUT "VOULEZ VOUS RECOMMENCER
   (OUI = 1 / NON = 2) ?"; RE
201 IF RE = 1 THEN 5
202 IF RE = 2 THEN FLASH :
   PRINT "AU REVOIR ET MERCI" : NORMAL
  : GOTO 230
205 GOTO 200
210 PRINT "TICK...TICK...FZZZZ...CLICK..."
215 PRINT
220 PRINT "VOUS AVEZ REUSSI"
225 PRINT : PRINT
   "AVEC VOS DOLLARS VIVE LES VACANCES"
226 PRINT : PRINT "AU BAHAMAS": GOTO 200
230 END
```

## EXÉCUTION

```
§RUN

     LE CAMBRIOLEUR
VOUS ETES UN CAMBRIOLEUR.
VOUS RECHERCHEZ UN CODE SECRET
PERMETTANT D'OUVRIR UN COFFRE-FORT
BIEN GARNI.
VOUS DEVEZ TROUVEZ LE CODE SECRET QUE
CONNAIT VOTRE ORDINATEUR.
ENTREZ UNE LETTRE DE VOTRE CHOIX ET IL
VOUS INDIQUERA SI ELLE SE TROUVE PLACEE
DANS L'ALPHABET AVANT OU APRES LE BON
CODE
VOUS DISPOSEZ DE 5 ESSAIS AVANT QUE LA
POLICE NE VOUS ARRETE
TAPER RETURN POUR CONTINUER

ESSAIS NO : 1

?L
LE BON CODE EST PLUS

LOIN QUE : L

ESSAIS NO : 2

?V
LE BON CODE EST PLUS

LOIN QUE : V

ESSAIS NO : 3

?X
LE BON CODE EST PLUS

LOIN QUE : X

ESSAIS NO : 4

?Z
TICK...TICK...FZZZZ...CLICK...

VOUS AVEZ REUSSI

AVEC VOS DOLLARS VIVE LES VACANCES

AU BAHAMAS

VOULEZ-VOUS RECOMMENCER
(OUI = 1 / NON = 2) ?2
AU REVOIR ET MERCI
```